

DENSO



Barcode Handy Terminal

BHT-700-CE

API Reference Manual

Copyright © DENSO WAVE INCORPORATED, 2007

All rights reserved. No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

Specifications are subject to change without prior notice.

All products and company names mentioned in this manual are trademarks or registered trademarks of their respective holders.

Contents

Chapter 1.	Software Requirements for the BHT-700	1
1.1.	Operating System (OS) on the BHT-700	1
1.2.	Application Development Software on the PC	1
1.2.1.	Application Development Tool	1
1.2.2.	Software Development Kit	1
Chapter 2.	Application Development Environment	2
2.1.	Required Hardware (PC to be used for application development)	2
2.2.	Required Software	2
2.3.	Installation	2
Chapter 3.	Output to the LCD Screen	3
3.1.	Screen Fonts	3
3.2.	Screen Rotation	3
3.2.1.	Setting the Screen Rotation Control Key	3
3.3.	Setting the Screen Rotation Angle	4
Chapter 4.	Backlight Control	5
4.1.	Outline	5
4.2.	Setting the Backlight Function On/Off Key	5
4.3.	Setting the Backlight Illumination Time	6
4.4.	Setting the Backlight Brightness and Power Saving Mode	6
4.5.	Controlling the Backlight with the Backlight Control Key	7
4.6.	Controlling the Backlight with the Backlight Control Function	8
4.7.	Key Backlight	9
Chapter 5.	Beeper and Vibrator Control	10
5.1.	Outline	10
5.2.	Setting the Beeper/Vibrator	11
5.3.	Starting/Stopping the Beeper/Vibrator	12
5.4.	Priority Orders between Events that Activate the Beeper/Vibrator	12
5.5.	Beeper Volume Patterns	12
Chapter 6.	Keys and Trigger Switch Control	13
6.1.	Outline	13
6.2.	Setting the Keys and Trigger Switch	14
6.3.	Shift Key Operation Mode	15
6.4.	Magic Key Control	15
6.5.	Assigning a User-Defined Key Code to the Magic Keys	16
6.5.1.	Assignment Method	16
6.5.2.	User-Defined Code Settings File (MKeyDef.txt)	16
6.6.	Key Input Modes	17
6.6.1.	Numeric Entry Mode	17
6.6.2.	Alphabet Entry Mode 1 (27 key pad)	17
6.7.	Function Mode	20
6.8.	Key Clicks	20
6.9.	Acquisition of Keypad Type	20
6.10.	Auto Repeat Function	21
Chapter 7.	LCD Status Indication	22
7.1.	Outline	22
7.2.	Setting the LCD Status Indication	23
Chapter 8.	Power Management	24
8.1.	Outline	24
8.2.	Standby	25
8.2.1.	Switching to the Standby State	25
8.2.2.	Standby Transition Prohibited Events	25
8.2.3.	Setting the Standby Transition Timeout	25
8.3.	Suspend	26
8.3.1.	Setting the Standby Transition Timeout	26
8.3.2.	Suspend Transition Prohibited Events	26
8.3.3.	Setting the Auto Power-off Timeout	26
8.3.4.	Setting the Effective Held-down Time of the Power Key for Switching to the Suspend State	26
Chapter 9.	Battery State	27

9.1.	Outline.....	27
9.2.	Battery Voltage Acquisition	27
9.3.	Battery Voltage Icon	27
9.4.	Battery Voltage Warning	27
Chapter 10.	LED	28
10.1.	Outline.....	28
10.2.	LED Control.....	28
10.2.1.	Display LED	28
10.2.2.	Charge LED	28
Chapter 11.	Data Communication	29
11.1.	Outline.....	29
11.2.	Programming for Data Communication	29
11.3.	Assigning Port Number	29
11.4.	ActiveSync	30
11.4.1.	Establishing an ActiveSync Connection	30
11.4.2.	ActiveSync Auto Connection Setting Method	30
Chapter 12.	Wireless Communication.....	31
12.1.	Outline.....	31
12.1.1.	Spread Spectrum Communications Method	31
12.1.2.	Configuration of Spread Spectrum System	31
12.2.	Programming for Wireless Communication	32
12.2.1.	Wireless Communication Parameters	33
12.2.2.	Opening and Closing the Wireless Communications Device.....	36
12.2.3.	Checking Synchronization with the Access Point	37
Chapter 13.	Barcode Reading.....	38
13.1.	Outline.....	38
13.1.1.	Enable Reading.....	38
13.1.2.	Specify Options in the BHT_EnableBar Function	40
13.1.3.	Barcode Buffer	41
13.2.	Programming.....	42
13.2.1.	Code Mark.....	42
13.2.2.	Multiple Code Reading	42
13.2.3.	Read Mode of the Trigger Switch	42
13.2.4.	Generating a Check Digit of Barcode Data.....	42
13.2.5.	Controlling the Indicator LED and Beeper/Vibrator as a Confirmation of Successful Reading	43
13.2.6.	Reading Split QR Codes (Only for BHT-700Q)	43
13.3.	Barcode Reading Using the Virtual COM Port	44
13.3.1.	Outline.....	44
13.3.2.	Programming.....	44
13.3.3.	How to Use.....	44
Chapter 14.	Updating OS.....	45
Chapter 15.	System Functions.....	46
15.1.	If a System Parameter Value is DWORD	47
15.2.	If a System Parameter Value is a Character String	49
15.3.	System Parameter Values That Can be Set/Obtained	51
15.4.	Device Information Acquisition	57
Chapter 16.	Device Control Functions	58
16.1.	Barcode API	60
16.2.	Backlight API.....	101
16.3.	Battery API.....	103
16.4.	LED API	105
16.5.	Beeper/Vibrator API	110
16.6.	Wireless Communication API.....	115
16.7.	OS Updating API.....	131
16.8.	Bluetooth API	132
16.9.	Other APIs	135
Chapter 17.	Programming Using OCX (OLE Customer Control)	141
17.1.	System Requirements	141
17.2.	Installation	141
17.3.	Using OCX	141
17.4.	Scanner Control	144

17.4.1.	Properties	144
17.4.2.	Methods.....	145
17.4.3.	Event Callback Function.....	148
17.4.4.	Error Codes	149
17.4.5.	Coding Sample.....	149
17.5.	File Transfer Control	150
17.5.1.	Properties	150
17.5.2.	Methods.....	151
17.5.3.	Event Callback Functions	157
17.5.4.	Coding Sample.....	159
Appendix A.	Keyboard Arrangement, Virtual Key Codes and Character Codes.....	160
A.1.	27-key pad	160
A.1.1.	Keyborard Arrangement.....	160
A.1.2.	Virtual Key Codes and Character Codes	161
A.1.3.	Character Codes in Alphabet Entry Mode.....	163
A.2.	42-key pad	164
A.2.1.	Keyborard Arrangement.....	164
A.2.2.	Virtual Key Codes and Character Codes	166
Appendix B.	Differences between Older Unit.....	169

Chapter 1. Software Requirements for the BHT-700

1.1. Operating System (OS) on the BHT-700

The OS running on the BHT-700 is Microsoft Windows CE 5.0.

1.2. Application Development Software on the PC

1.2.1. Application Development Tool

The application development tool for the BHT-700 is Microsoft eMbedded Visual C++ 4.0 (Service Pack 4)

1.2.2. Software Development Kit

The BHT-700 Software Development Kit provides the application development environment for Windows CE set up on the BHT-700. It includes the following libraries:

(1) Help files

(2) Windows-CE standard header files

(3) Windows-CE standard library files

(4) BHT-dedicated header file : BHTLIB.h

- Includes statements for declaring BHT-dedicated APIs prototypes and macro definition of constants.
- To use the BHT-dedicated APIs, the BHTLIB.h should be included.

(5) BHT-dedicated library : BHTLIB.lib

- Includes BHT-dedicated barcode reading functions and device driver management functions.
- To use the BHT-dedicated APIs, the BHTLIB.lib should be linked.

(6) BHT-dedicated OCX files : Scanner700.ocx (for BHT-700B), Scanner700Q.ocx (for BHT-700Q), FileTransfer700.ocx, and FileTransferPC.ocx (for PC)

- Include BHT-dedicated barcode scanning functions and file transfer functions.
- To use the BHT-dedicated OCX, Scanner700.ocx, Scanner700Q.ocx, and FileTransfer700.ocx should be linked.

Chapter 2. Application Development Environment

2.1. Required Hardware (PC to be used for application development)

Item	Specification
OS	Microsoft Windows 2000 Professional with Service Pack 2 or higher, or Microsoft Windows 2000 Server with Service Pack 2 or higher, or Microsoft Windows XP Professional or higher.
PC	With a Pentium-II class processor, 450 MHz or faster
Memory	For Microsoft Windows 2000 Professional or Microsoft Windows XP Professional: 96 MB or more (128 MB or more recommended) ----- For Microsoft Windows 2000 Server : 192 MB or more (256 MB or more recommended)
HDD	200 MB or more hard disk space
Display	VGA or higher-resolution monitor. A Super VGA (800 x 600 or larger) monitor is recommended.

2.2. Required Software

Application development tool: Microsoft eMbedded Visual C++ 4.0 (SP4)

You can download Microsoft eMbedded Visual C++ 4.0 and Service Pack 4 from the Microsoft Web site:
(Microsoft eMbedded Visual C++ 4.0)

<http://www.microsoft.com/downloads/details.aspx?FamilyID=1dacdb3d-50d1-41b2-a107-fa75ae960856&DisplayLang=en>

(Service Pack 4)

<http://www.microsoft.com/downloads/details.aspx?FamilyID=4a4ed1f4-91d3-4dbe-986e-a812984318e5&displaylang=en>

APIs available for eMbedded Visual C++ are:

- (1) Win32API
- (2) Microsoft Foundation Class (MFC)
- (3) Dedicated APIs (for device control or data entry from the BHT)

Software development kit: BHT700_XXX_SDK.msi

This should be embedded into Microsoft eMbedded Visual C++ 4.0 for use.

2.3. Installation

The Microsoft eMbedded Visual C++ 4.0 and BHT-700 software development kit should be installed to an application development PC in this order. To install the development kit, run the BHT700_XXX.msi in the BHT-700 Software Development Kit CD.

Chapter 3. Output to the LCD Screen

3.1. Screen Fonts

The BHT-700 has the following integrated screen fonts:

- (1) Arial (ttf)
- (2) Courier New (ttf)
- (3) Tahoma (ttf)
- (4) Time New Roman (ttf)
- (5) Wingding (ttf)

If no screen font is specified, Tahoma applies automatically.

3.2. Screen Rotation

The screen can be rotated using either of the following methods.

- (1) By pressing the screen rotation control key.
- (2) By using the system setting function (**BHT_SetSysSettingDW**).

3.2.1. Setting the Screen Rotation Control Key

The screen rotation control key can be set using the

BHT_SetSysSettingDW (BHT_DISP_ROTATION_KEY,...) function.

Furthermore, the setting can be obtained using the

BHT_GetSysSettingDW (BHT_DISP_ROTATION_KEY,...) function.

The relationship between the settable screen rotation control keys and settings is outlined in the following table.

Screen Rotation Control Key	Set value	Screen Rotation Control Key	Set value
[M1]	0x00000201	[SF]+[M1]	0x00010201
[M2]	0x00000202	[SF]+[M2]	0x00010202
[M3]	0x00000203	[SF]+[M3]	0x00010203

3.3. Setting the Screen Rotation Angle

The settable rotation angles are 0°, 90°, 180°, and 270°. The direction is anti-clockwise.

The screen rotation angle can be set and read using the

BHT_SetSysSettingDW (DWORD dwCtrlCode, DWORD dwSysParam) and

BHT_GetSysSettingDW (DWORD dwCtrlCode, DWORD *pdwSysParam) functions, respectively.

Parameter	Type	R/W	Control Code (dwCtrlCode)	Parameter Value (dwSysParam)	Default	Validation Timing
Screen rotation angle	DW	R/W	BHT_DISP _ROTATION	DISP_ROTATION_0 : 0° DISP_ROTATION_90 : 90° DISP_ROTATION_180 : 180° DISP_ROTATION_270 : 270°	DISP_ROTATION_0	Immediately after setting

Chapter 4. Backlight Control

4.1. Outline

The backlight illumination and power saving modes can be controlled using either of the following methods.

- (1) The backlight can be controlled by pressing the backlight control key.
- (2) The backlight can be controlled using the backlight control function (**BHT_SetBlStatus**).

The following backlight related setting items are also available.

- (1) Backlight control key
- (2) Backlight illumination time
- (3) Backlight brightness
- (4) Backlight power saving mode

Furthermore, the BHT-700 keypad is also equipped with a backlight (hereafter referred to as key backlight) for which the following settings can be made.

- (1) Illumination device (when **BHT_SetBlStatus** is called)
- (2) Key backlight illumination trigger

4.2. Setting the Backlight Function On/Off Key

You can assign the backlight function on/off key to other keys by the **BHT_SetSysSettingDW** (**BHT_BACKLIGHT_KEY...**) function or by assigning the backlight control function to the magic key. The table below lists the relationship between the keys that act as a backlight function on/off key and the set values in the **BHT_SetSysSettingDW** (**BHT_BACKLIGHT_KEY...**) function.

If no key is specified as a backlight function on/off key, the combination of the SF key and M3 key works as a backlight function on/off key by default.

Backlight control key	Set value	Backlight control key	Set value
[M1]	0x00000201	[SF]+[M1]	0x00010201
[M2]	0x00000202	[SF]+[M2]	0x00010202
[M3]	0x00000203	[SF]+[M3]	0x00010203

[Ex]

Execute function **BHT_SetSysSettingDW** (**BHT_BACKLIGHT_KEY**, 0x00010201) when assigning a simultaneous combination of the [SF] and [M1] keys to the backlight control key.

4.3. Setting the Backlight Illumination Time

The backlight illumination time is set and read using the **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions.

Parameter	Type	R/W	Control Code (dwCtrlCode)	Parameter Value (dwSysParam)	Default	Validation Timing
Illumination time when powered by battery (sec.)	DW	R/W	BHT_BACKLIGHT_BATT_TIME	0 - 255 0: Backlight OFF 255: Continuously ON	3	When backlight illumination timer is next reset
Illumination time when placed on CU (sec.)	DW	R/W	BHT_BACKLIGHT_AC_TIME	0 - 255 0: Backlight OFF 255: Continuously ON	60	When backlight illumination timer is next reset

4.4. Setting the Backlight Brightness and Power Saving Mode

The backlight brightness and power saving mode are set and read using the **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions.

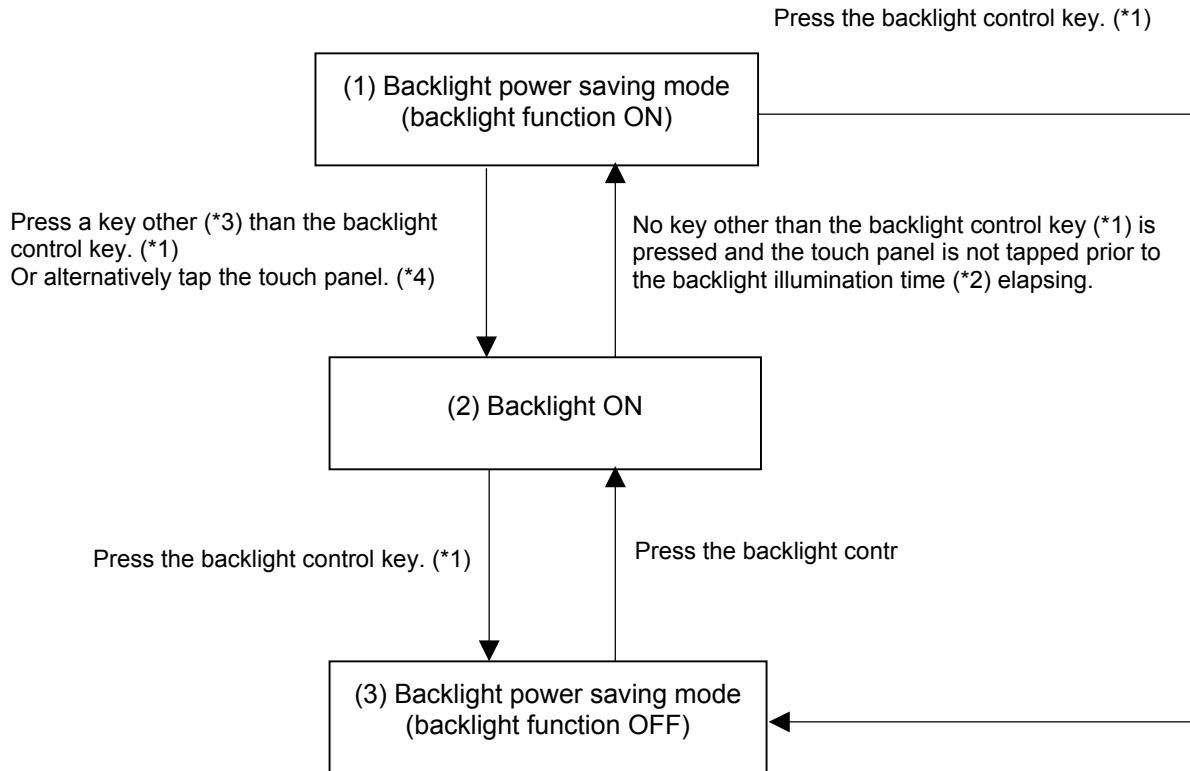
Parameter	Type	R/W	Control Code (dwCtrlCode)	Parameter Value (dwSysParam)	Default	Validation Timing
Backlight brightness	DW	R/W	BHT_BACKLIGHT_BRIGHTNESS	0: OFF 1: Dark 2: Bright (low) 3: Bright (high)	3	When the backlight is next turned ON
Backlight power saving mode	DW	R/W	BHT_BACKLIGHT_POWERSAVE	0: OFF 1: Dim	1	When backlight illumination status is set to power saving mode first after setting

4.5. Controlling the Backlight with the Backlight Control Key

The backlight function can be enabled/disabled by pressing the backlight function control key (Default: Hold down [SF] key and press [M3]).

The illumination time is specified using the **BHT_SetSysSettingDW**

(BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME, ...) function. The default value is 3 seconds when powered by the battery, and 60 seconds when placed on the CU. Backlight control is performed as shown in the flow diagram below.



(*1)

Default: Hold down [SF] key and press [M3].

Setting is possible using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY,...) function.

(*2)

The backlight illumination time is set using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME,...) function. Power saving mode is enabled if no key other than the backlight control key is pressed, or if the touch panel is not tapped within this time.

This time is measured from the point all keys are released or the touch panel is last pressed.

(*3)

If key-press has not been set for the key backlight illumination trigger, the key backlight will not illuminate even if a key is pressed. If, however, the key backlight is already illuminated beforehand, it will not turn OFF by pressing a key.

(*4)

If touch panel tap has not been set for the key backlight illumination trigger, the key backlight will not illuminate even if the touch panel is tapped. If, however, the key backlight is already illuminated beforehand, it will not turn OFF by tapping the touch panel.

(*5)

Cold booting is performed from the status at (1) above.

However, cold booting is performed from the status at (1) when the registry is saved with the status at (1) or (2), and is performed from the status at (3) when the registry is saved with the status at (3).

(*6)

When performing warm booting or when resuming from the suspend status, the process is performed from (1) if the status prior to warm boot/suspend is (1) or (2), and is performed from (3) if the status prior to warm boot/suspend is (3).

4.6. Controlling the Backlight with the Backlight Control Function

The backlight function can be controlled using the **BHT_SetBltStatus** function.

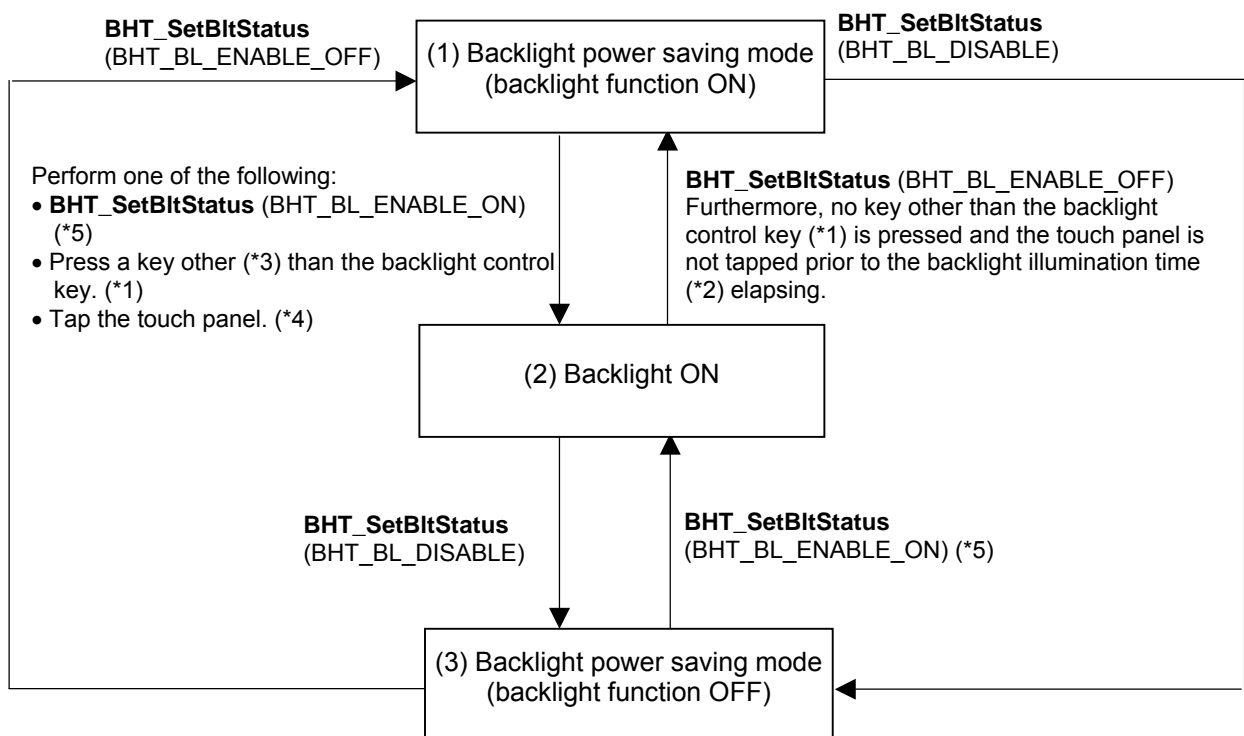
The **BHT_SetBltStatus** (BHT_BL_ENABLE_ON) function is used to enable the backlight function and turn the backlight ON.

The backlight power saving mode is enabled if no keys are pressed, or the touch panel tapped from the point the backlight is turned ON using the **BHT_SetBltStatus** (BHT_BL_ENABLE_ON) function until the time set using the **BHT_SetSysSettingDW**

(BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME,...) function (Default: 3 seconds when powered by battery, 60 seconds when placed on CU) elapses, or if the **BHT_SetBltStatus** (BHT_BL_ENABLE_OFF) function is executed. (The backlight function remains ON at this time.)

If the **BHT_SetBltStatus** (BHT_BL_DISABLE) function is executed, the backlight function is disabled, and the backlight power saving mode is enabled.

Backlight control is performed as shown in the flow diagram below.



(*1)

Default: Hold down [SF] key and press [M3].

Setting is possible using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY,...) function.

(*2)

The backlight illumination time is set using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME,...) function. Power saving mode is enabled if no key other than the backlight control key is pressed, or if the touch panel is not tapped within this time.

This time is measured from the point all keys are released or the touch panel is last pressed.

(*3)

If key-press has not been set for the key backlight illumination trigger, the key backlight will not illuminate even if a key is pressed. If, however, the key backlight is already illuminated beforehand, it will not turn OFF by pressing a key.

(*4)

If screen tap has not been set for the key backlight illumination trigger, the key backlight will not illuminate even if the screen is tapped. If, however, the key backlight is already illuminated beforehand, it will not turn OFF by tapping the screen.

(*5)

The backlight specified with the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_DEVICE,...) function illuminates.

(*6)

Cold booting is performed from the status at (1) above.

However, cold booting is performed from the status at (1) when the registry is saved with the status at (1) or (2), and is performed from the status at (3) when the registry is saved with the status at (3).

(*7)

When performing warm booting or when resuming from the suspend status, the process is performed from (1) if the status prior to warm boot/suspend is (1) or (2), and is performed from (3) if the status prior to warm boot/suspend is (3).

4.7. Key Backlight

The following settings can be made for the key backlight.

Parameter	type	R/W	Control code (dwCtrlCode)	Parameter value (dwSysParam)	Default	Validating timing
Device illuminated when BHT_SetBlitStatus called	DW	R/W	BHT_BACKLIGHT_DEVICE	One of the following combinations: 0: None LIGHTING_LCD(=1) : LCD LIGHTING_KEY(=2) : KEY LIGHTING_LCD LIGHTING_KEY(=3) : Both	1	Immediately after setting, when BHT_SetBlitStatus next called
Key backlight illumination trigger	DW	R/W	BHT_BACKLIGHT_KEY_FACTOR	0 : Always OFF BHT_BLT_KEY_FACTOR_KEY : Illuminate only when keys pressed BHT_BLT_KEY_FACTOR_KEYTAP : Illuminate when keys pressed or tapped	1	Immediately after setting, first tap or key press when "BHT_BLT_KEY_FACTOR_KEY" or "BHT_BLT_KEY_FACTOR_KEYTAP"

Chapter 5. Beeper and Vibrator Control

5.1. Outline

The beeper and vibrator are controlled by:

- (1) the beeper/vibrator setting functions
(that allow you to choose beeper and/or vibrator and set the beeper volume. Refer to Section 5.2.)
- (2) the beeper/vibrator start/stop functions
(that allow you to set the beeping or vibration interval, the number of repetitions, and frequency. Refer to Section 5.3.)

5.2. Setting the Beeper/Vibrator

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the beeper/vibrator parameters as specified below.

Parameter name	Type	R/W	Control code (dwCtrlCode)	Parameter value (dwSysParam)	Default	Validating timing
Rumble device	DW	R/W	BHT_BEEP_VIB_SELECT	BEEP_SELECT : Beeper VIB_SELECT : Vibrator BEEP_SELECT VIB_SELECT : Beeper and vibrator	BEEP_SELECT	First sound after setting
Beeper volume (*1)	DW	R/W	BHT_BEEP_VIB_VOLUME	0: OFF 1 (Lowest) to 5 (Highest)	5	First sound after setting
Key clicks (*2)	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1 (Soft) 2 (Loud)	2	First sound after setting
Screen taps	DW	R/W	BHT_BEEP_VIB_TAP	0: OFF 1 (Soft) 2 (Loud)	2	First sound after setting
Trigger switch clicks (*3)	DW	R/W	BHT_BEEP_VIB_TRGKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Permit	CLICK_SOUND_OFF	First sound after setting

(*1) This setting is effective only when the value 0, 1, or 2 is specified to the frequency in the beeper start/stop functions (**BHT_StartBeep** or **BHT_StartBeeperOnly**).

(*2) This excludes the pressing of magic keys assigned to the trigger switch and the [SCAN] key when the "trigger switch click sound" is OFF.

(*3) This is effective only when pressing magic keys assigned to the trigger switch and the [SCAN] key.

The rumble device specification above takes effect when the beeper/vibrator is driven:

- (1) by the **BHT_StartBeep** function.
- (2) due to low battery warning, in conjunction with the "Battery voltage has lowered." or "Charge the Battery!" message.
- (3) upon completion of barcode reading.

The MessageBox, MessageBeep and PlaySound Windows CE standard APIs and Windows CE standard warning and notification sounds are enabled by the audio function, and therefore there is no influence on settings made with the above functions.

The sound pattern of the key clicks, screen taps, and trigger switch clicks is as follows:

ON-duration: 10 ms
Frequency: 1396 Hz
Volume : Loud, Soft

5.3. Starting/Stopping the Beeper/Vibrator

The beeper/vibrator is activated or deactivated by the following functions:

Function	Used to:
BHT_StartBeep	Activate the selected device (beeper or vibrator).
BHT_StartBeeperOnly	Activate the beeper.
BHT_StartVibratorOnly	Activate the vibrator.

The functions listed above start the beeper/vibrator control and immediately pass control to the subsequent statement or function. The actual device operation is carried out in background processing.

Specifying the frequency with value 0, 1, or 2 sounds the beeper with the frequency listed below. If any other value is specified, the beeper sounds at the maximum volume.

Parameter value	Frequency (Hz)
0	698
1	1396
2	2793

If the suspend or critical power states are turned OFF while the beeper is sounding or the vibrator is vibrating, the BHT resumes with both the beeper and vibrator stopped when the unit is next resumed.

5.4. Priority Orders between Events that Activate the Beeper/Vibrator

There are priority orders between events that activate the beeper/vibrator as listed below.

Priority	Event that activate the beeper/vibrator
Higher ↑ ↓ Lower	System error
	Completion of barcode reading
	Setting in applications
	Key clicks or screen taps

When the beeper or vibrator is being driven by any event, the lower priority event (if happens) activates no beeper or vibrator but the same or higher priority event (if happens) overrides the currently operating beeper or vibrator and newly activates the beeper or vibrator.

5.5. Beeper Volume Patterns

The beeper is activated according to the beeper volume as listed below.

Beeper volume	Volume
1 (lowest)	Soft
2	
3	Mid
4	
5 (highest)	Loud

Chapter 6. Keys and Trigger Switch Control

6.1. Outline

In addition to the processing for depressed or released keys and trigger switch, the BHT OS controls the following functions assigned to them.

- (1) Specifying the shift key operation mode
- (2) Assigning functions to the magic keys (M1 to M3)
- (3) Supporting the alphabet entry mode (in addition to the numeric entry mode)
- (4) Function mode
- (5) Key click sound
- (6) Keyboard type acquisition

Furthermore, both the 27-key pad and 42-key pad keyboard types are supported.

6.2. Setting the Keys and Trigger Switch

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam)

and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the keys and trigger switch parameters.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Shift key operation mode	DW	R/W	BHT_KEY_SHIFT_MODE	KEY_NON_LOCK : Non-lock mode KEY_ONE_TIME : Onetime lock mode	KEY_NON_LOCK	Immediately after setting
Assignment to M1 key	DW	R/W	BHT_KEY_M1_MODE	MAGIC_FUNC_NONE : Ignore the depressed key	MAGIC_FUNC_TAB	Immediately after setting
Assignment to M2 key	DW	R/W	BHT_KEY_M2_MODE	MAGIC_FUNC_ENTER : Treat as ENT key	MAGIC_FUNC_NONE	Immediately after setting
Assignment to M3 key	DW	R/W	BHT_KEY_M3_MODE	MAGIC_FUNC_TRG : Treat as trigger switch MAGIC_FUNC_SHIFT : Treat as SF key MAGIC_FUNC_ALT : Treat as ALT key MAGIC_FUNC_CTRL : Treat as CTRL key MAGIC_FUNC_BLT : Treat as backlight function on/off key MAGIC_FUNC_TAB : Treat as TAB key MAGIC_FUNC_CLEAR : Treat as CLEAR key MAGIC_FUNC_USERDEF : User-defined code (*1)	MAGIC_FUNC_TRG	Immediately after setting
Entry mode	DW	R/W	BHT_KEY_INPUT_METHOD	INPUT_METHOD_NUMERIC : Numeric entry mode INPUT_METHOD_ALPHABET : Alphabet entry mode 1 INPUT_METHOD_ALPHABET 2 : Alphabet entry mode 2 (*2)	27-key type: INPUT_METHOD_NUMERIC 42-key type: INPUT_METHOD_ALPHABET	Immediately after setting
Enable/disable alphabet entry switching key	DW	R/W	BHT_DISABLE_KEYMODE_CHANGE_KEY	ENABLE_KEY_TOCHANGE_ALPHABET : Enable alphabet entry DISABLE_KEY_TOCHANGE_ALPHABET : Disable alphabet entry	ENABLE_KEY_TOCHANGE_ALPHABET	Immediately after setting
Function mode	DW	R/W	BHT_KEY_FUNCTION	KEY_FUNCTION_ON : Function mode KEY_FUNCTION_OFF : Non-function mode	KEY_FUNCTION_OFF	Immediately after setting

(*1) User-defined codes can only be acquired.

(*2) Alphabet entry mode 2 is only available with the 27-key pad.

6.3. Shift Key Operation Mode

The shift key operation mode works as follows:

Shift key operation mode	Description
Non-lock mode	- The keypad is shifted when the Shift key is held down.
Onetime lock mode	- The shift status is cleared immediately after releasing a key when in the shift status from the time the key is pressed until it is released while the shift key is held down and after it is released.

6.4. Magic Key Control

The table below lists the virtual key codes and character codes when magic keys (M1 to M3) are pressed.

Parameter value	Virtual key code			Character code	
	Constant		Value	When not shifted	Shifted
MAGIC_FUNC_NONE	[M1] key	VK_M1	C1	-	-
	[M2] key	VK_M2	C2	-	-
	[M3] key	VK_M3	C3	-	-
MAGIC_FUNC_ENTER	VK_RETURN		0D	0D(CR)	0D(CR)
MAGIC_FUNC_TRG	(*1)			-	-
MAGIC_FUNC_SHIFT	VK_SHIFT		10	-	-
MAGIC_FUNC_CTRL	VK_CONTROL		11	-	-
MAGIC_FUNC_ALT	VK_MENU		12	-	-
MAGIC_FUNC_BLT	(*1)			-	-
MAGIC_FUNC_TAB	VK_TAB		09	09 (tab)	09 (tab)
MAGIC_FUNC_LASER	(*1)			-	-
MAGIC_FUNC_CLEAR	VK_CLEAR		0C	-	-
MAGIC_FUNC_USERDEF	(*2)				

(*1) Returns the same virtual key code as when "MAGIC_FUNC_NONE" is assigned.

(*2) Notified virtual key codes are user-defined codes. In such a case, functions assigned to keys such as the following cannot be executed.

- Changing the entry mode by pressing the [AL] key to which the function has been assigned.
- Backing up the registry by pressing the [SF] key to which the function has been assigned and the [Power] key.
- Switching between backlight enable and backlight disable by assigning the function to [SF] + [M3].

6.5. Assigning a User-Defined Key Code to the Magic Keys

Apart from the previously mentioned functions, optional keys can be applied to the magic keys following the method below.

With this function it is possible to assign keys to the magic keys that do not exist in the BHT-700, or to execute the equivalent of a multi-key function by pressing a magic key once.

6.5.1. Assignment Method

The steps for setting user-defined key codes for the magic keys are as follows:

- (1) Save a user-defined code settings file with the filename "MKeyDef.txt" in the FLASH folder of the BHT.
 - (2) Choose the key you wish to set from the key definition menu in the BhtShell (for further details refer to the "BHT-700BB/700BWB/700BWBG-CE User's Manual" or "BHT-700QWBG-CE User's Manual").
- Backup files can be created with a backup registry.

6.5.2. User-Defined Code Settings File (MKeyDef.txt)

- (1) File name
"MKeyDef.txt" (fixed)
- (2) Format
<Character string inside the combo box>,<Defined code number>,<Defined code 1>,...,<Defined code 4>

Item	Display Method	Setting Content
Character string inside the combo box	Character string	A character string containing up to 64 characters. Extra characters will be ignored.
Defined code number	decimal number	A user-defined code specified as a number between 1 and 4.
Defined code 1 through 4	hexadecimal number	The virtual key code you wish to assign.

[Ex] Setting a user-defined key code of "Alt + X" and "Alt + Y" to be added to the combo box list.

ALT+X, 2, 0x12, 0x58
ALT+Y, 2, 0x12, 0x59

- (*) If there is a mistake in the format of a line in the MKeyDef.txt file, that line will be ignored and removed from the BhtShell key definition menu.
- (*) Even if the MKeyDef.txt file is deleted, key code settings will be retained (the BhtShell will display "None"). When a different function is designated in the BhtShell, the previous key code settings will be replaced.

6.6. Key Input Modes

The BHT-700 key pad has the following three key entry modes.

(1) Numeric entry mode

This mode allows you to type in numeric data with the numeric keys.

(2) Alphabet entry mode 1

In the 27-key pad, use the numeric keys to type in alphabet letters in the same way as he/she uses a cellular phone.

In the 42-key pad, use the alphabet keys to type in alphabet letters directly.

(3) Alphabet entry mode 2 (27-key pad only)

This entry mode is for alphabet entry at programs running on the computer when using terminal services such as a remote desktop connection.

Similarly to alphabet entry mode 1, alphabet letters are entered using the numeric keys.

6.6.1. Numeric Entry Mode

The numeric entry mode starts by:

(1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_NUMERIC) function.

(2) pressing the [AL] key when in alphabet entry mode 2 (27-key pad). (*1), or pressing the [NUM] key when in alphabet entry mode 1 (42-key pad). (*1)

(*1) Effective only when the key entry mode transition key is not disabled.

Pressing keys in this mode returns virtual key codes and character codes specified in Appendix A.

6.6.2. Alphabet Entry Mode 1 (27 key pad)

The alphabet entry mode 1 starts by:

(1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_ALPHABET) function.

(2) pressing the AL key(*2) in the numeric entry mode.

The alphabet entry mode 1 terminates by:

(1) switching to any other entry mode with the **BHT_SetSysSettingDW** function.

(2) pressing the AL key(*2) in the numeric entry mode.

(*2) The key takes effect only when it is not disabled by the BHT_DISABLE_KEYMODE-CHANGE_KEY.

In the 27-key pad alphabet entry mode 1, alphabet characters can be entered using an alphabet character similar to that used on a cellular phones.

(1) When changing to alphabet entry mode 1, an unestablished character display window similar to that shown below displays.



Unestablished characters display.

The unestablished character display window has the following features.

- This window can be moved by using the stylus.
- The focus is not transferred to the unestablished character display window.
- The unestablished character display window always displays in the foreground.

Furthermore, the following icon displays in the task bar when in alphabet entry mode 1.



- (2) If keys [0] to [9] or the [.] key is pressed, the pressed key becomes an unestablished character and displays in the unestablished character display window. The character then reverts to a character code when any of these keys becomes established.

Press any of the following keys below to establish unestablished characters.

- Keys [0] to [9] or [.] that differ from the key pressed at the unestablished character
- [ENT] key
- "MAGIC_FUNC_ENTER" assigned to the magic keys
- Keys [F1] to [F12]

- (3) Keys used for alphabet entry 1

The table below lists keys whose operations are different from those in the numeric entry mode.

Use this key	To do this
0 to 9 and period (.) keys	Enter alphabets. For alphabets assigned to these keys, refer to "Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes" – "A.1.3. Character Codes in Alphabet Entry Mode."
ENT key	Establish an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
BS key	Clear an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
F1 to F12 Key	Establish an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
Magic key	Establish an unestablished key if any when the MAGIC_FUNC_ENTER is assigned to these keys. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
AL key	Clears unestablished keys if any exist and switches to numeric entry mode.

6.6.3. Alphabet Entry Mode 1 (42-Key Pad)

Alphabet entry mode 1 starts by:

- (1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_ALPHABET) function.

- (2) pressing the [NUM] key when in numeric entry mode. (*2)

Alphabet entry mode 1 terminates by:

- (1) switching to any other entry mode with the **BHT_SetSysSettingDW**(BHT_KEY_INPUT_METHOD, INPUT_METHOD_XXXXXX) function.

- (2) pressing the [NUM] key. (*2)

(*2) Effective only when the key entry mode transition key is not disabled.

When keys are pressed in this mode, virtual key codes and character codes are returned in accordance with "Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes"

–"A.2.2. Virtual Key Codes and Character Codes".

6.6.4. Alphabet Entry Mode 2 (27-key pad only)

Alphabet entry mode 2 starts by:

- (1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_ALPHABET2) function.
- (2) pressing the [AL] key when in alphabet entry mode 1. (*1)

Alphabet entry mode 2 terminates by:

- (1) switching to any other entry mode with the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_XXXXXX) function.
- (2) pressing the [AL] key. (*1)
(*1) Effective only when the key entry mode transition key is not disabled.

Similarly to alphabet entry mode 1, alphabet letters can also be entered in alphabet entry mode 2 using the same method used when entering alphabet letters with a cellular phone.

- (1) The following icon below displays in the task bar when starting alphabet entry mode 2.
The unestablished character display window does not display.



- (2) If keys [0] to [9] or the [.] key is pressed, characters assigned to those keys display at the current cursor position. By pressing the same key(s) again, assigned characters display sequentially. Press a different key to establish the entered character(s).

- (3) Keys used in alphabet entry mode 2

The table below lists keys whose operations are different from those in the numeric entry mode.

Use this key	To do this
[0] to [9] and period (.) keys	Used to enter alphabet letters. For details of alphabet letters assigned to these keys, refer to “Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes” – “A.1.3. Character Codes in Alphabet Entry Mode.”
[AL] key	Switches to numeric entry mode.

6.7. Function Mode

Use either of the methods below to enable function mode.

- (1) Call up the **BHT_SetSysSettingDW** (BHT_KEY_FUNCTION,KEY_FUNCTION_ON) function.
- (2) Press the [FN] key when in function mode.

Use either of the methods below to disable function mode and return to non-function mode.

- (1) Call up the **BHT_SetSysSettingDW** (BHT_KEY_FUNCTION,KEY_FUNCTION_OFF) function.
- (2) Press the [FN] key when in function mode.

Non-function mode is enabled as the default when the unit is booted up.

The following icon displays in the task bar when in function mode.



If a key is pressed when in function mode, a virtual key code or character code is returned as outlined in "Appendix A. Keyboard Arrangement, Virtual Key Codes, and Character Codes".

6.8. Key Clicks

When the keys are pressed, the BHT clicks as specified below. Note that pressing the power key does not click.

Parameter name	Type	R/W	Control code (dwCtrlCode)	Parameter value (dwSysParam)	Default	Validating timing
Key click volume	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1: Soft 2: Loud	2	first key press after setting
Trigger switch clicks	DW	R/W	BHT_BEEP_VIB_TRGKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Allow	CLICK_SOUND_OFF	first trigger key press after setting

6.9. Acquisition of Keypad Type

The **BHT_GetSysSettingDW** (DWORD dwCtrlCode,DWORD *pdwSysParam) function reads the keypad type.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Keypad type	DW	R	BHT_KEYBOARD_TYPE	KEYBOARD_TYPE1 : 27-key pad KEYBOARD_TYPE2 : 42-key	-	-

6.10. Auto Repeat Function

The keys used to perform auto repeat are listed in the following table. Whether auto repeat function for each key is enabled or disabled is listed in the following table.

Key	27-key pad	42-key pad
[0] to [9] key and Period ([.]) key	- In Numeric entry mode : ● - In Alphabet entry mode : -	●
[A] to [Z] key	-	●
[BS] key	●	●
[C] key	●	●
[◀][▶][▲][▼] key	●	●
[F1] to [F12] key	●	●
[SF] key	-	-
[FN] key	-	-
[ENT] key	-	-
[TAB] key	●	●
[AL] key	-	N/A
[NUM] key	N/A	-
[ESC] key	-	-
[SCAN] key	-	-
Magic keys	- No key assignment : - - [ENT]key : - - Trigger key : - - Shift key : - - Backlight control key : - - [TAB] key : ● - [CTRL] key : - - [ALT] key : - - [CLEAR] key : - - User-defined code : -	- No key assignment : - - [ENT] key : - - Trigger key : - - Shift key : - - Backlight contorol key : - - [TAB] key : ● - [CTRL] key : - - [ALT] key : - - [CLEAR] key : - - User-defined code : -
Power key	-	-

* ● : Auto repeat performed , - : Auto repeat not performed

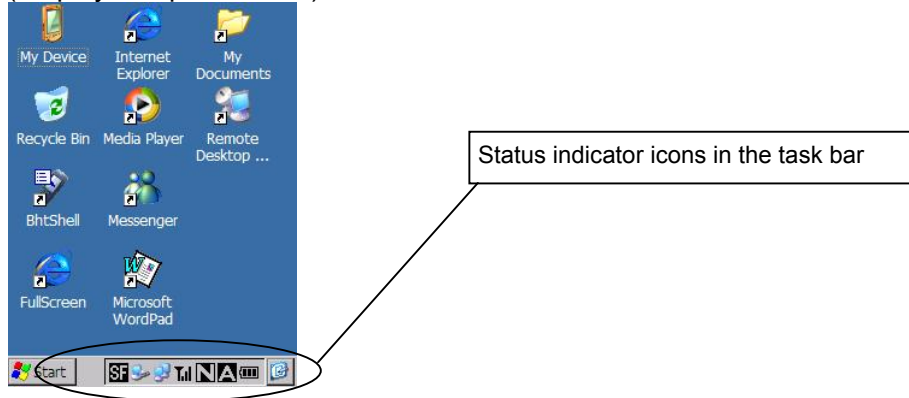
Chapter 7. LCD Status Indication

7.1. Outline

The status of the BHT is displayed on the LCD as specified below.

Status	Description	Icon
Battery voltage level	Displays the battery voltage in five levels.	
Software keyboard display state	Shows whether the software keyboard is displayed or hidden. Tapping this icon toggles the software keyboard on and off.	: The software keyboard is displayed. : The software keyboard is hidden.
Keypad shift state	Displays the icon when the keypad is shifted.	
Function state	Displays the icon when in function mode.	
Alphabet input state (27-key pad only)	Displays the ALP window when the alphabet input function is activated. An unestablished character appears in this ALP window.	
	Displays the icon when the alphabet input function is activated.	
Numeric entry status (42-key pad only)	Displays when in numeric entry mode.	
Standby state	Displays this icon when the CPU comes to be on standby.	
Synchronization state	Displays the open state of the wireless device and the radio field intensity.	<p>The wireless device is open.</p> <p>The wireless device is open and the wireless link is established with an access point.</p> : Radio field intensity (Low) : Radio field intensity (Medium) : Radio field intensity (High)
ActiveSync	Displays this icon when the BHT is communicating with the PC via Microsoft ActiveSync (not using LAN).	
Desktop display	Switches the screen between the application execution display and desktop display. Tapping this icon when an application program is running switches the screen to the desktop display. Tapping it again returns to the application execution display.	
Bluetooth power status	Displays the Bluetooth power status. No icons display if the unit is not equipped with a Bluetooth device.	: Power ON : Power OFF

(Display sample of icons)



7.2. Setting the LCD Status Indication

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the LCD status indication as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Battery voltage level icon	DW	R/W	BHT_ICON_BATTERY	0: Hide 1: Display	1	Immediately after setting
Software keyboard icon	DW	R/W	BHT_ICON_SIP	0: Hide 1: Display	1	Immediately after setting
Keypad shift icon	DW	R/W	BHT_ICON_SHIFTKEY	0: Hide 1: Display	1	Immediately after setting
Alphabet input icon(27-key pad only)	DW	R/W	BHT_ICON_IN_ALPHA	0: Hide 1: Display	1	Immediately after setting
Synchronization state icon	DW	R/W	BHT_ICON_RADIO_INTENSE	0: Hide 1: Display	1	Immediately after setting
Standby state icon	DW	R/W	BHT_ICON_STANDBY	0: Hide 1: Display	0	Immediately after setting
Function state icon	DW	R/W	BHT_ICON_FUNC	0: Hide 1: Display	1	Immediately after setting
Numeric entry status (42-key pad only)	DW	R/W	BHT_ICON_NUMERIC	0: Hide 1: Display	1	Immediately after setting
Bluetooth power status	DW	R/W	BHT_ICON_BLUETOOTH	0: Hide 1: Display	0	Immediately after setting

Chapter 8. Power Management

8.1. Outline

The power management functions switch the system powering state.

The following four system power states exist.

- (1) Power ON
- (2) Standby
- (3) Suspended (*1)
- (4) Critical OFF (*2)

(*1) Suspend

The BHT will be suspended when the power is turned off with the power key or auto power off feature.

(*2) Critical OFF

The BHT will become critical off when the power is turned off due to battery voltage drop or battery cover unlocked.

Notes

- No processing is performed when the BHT is on standby.
- When the SD memory card is used, disable the standby function before accessing the card.

8.2. Standby

8.2.1. Switching to the Standby State

The BHT switches from the power ON state to the standby state when any of the following conditions arises:

- (1) When the standby transition timeout occurs after a standby transition prohibited event (listed below) is completed.
- (2) When waiting for the event specified by the **BHT_WaitStandbyEvent** function with the standby transition prohibited event completed.
- (3) When the standby transition prohibited event is completed while waiting for the event specified by the **BHT_WaitStandbyEvent** function to occur.

8.2.2. Standby Transition Prohibited Events

The following items are standby transition prohibited events.

- Key being pressed
- Touch panel being tapped
- Screen being refreshed
- Beeper/vibrator in operation
- Key click sound/touch panel tap sound in operation
- Backlight being ON (excludes those times when continuously ON)
- Reading barcodes
- IrDA interface port opened
- Connector interface port opened
- USB interface opened
- Wireless device opened
- During USB-LAN communication
- Flash memory being erased or written
- RTC being accessed
- Indicator LED being ON
- System message being displayed
- Bluetooth device power being ON
- Explorer displayed
- Standby transition time set to "0"

8.2.3. Setting the Standby Transition Timeout

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the standby transition timeout as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Standby transition timeout (in units of 100 msec)	DW	R/W	BHT_PM_STBYTIME	0: Disable 1 - 255	10 (1 sec)	Immediately after setting

8.3. Suspend

8.3.1. Setting the Standby Transition Timeout

The BHT switches to the suspend state when any of the following conditions arises:

- (1) When the power is on, the power key is held down for the effective held-down time (for switching to the suspend state) or more.
- (2) An auto power-off timeout occurs after one of the suspend transition prohibited events (listed below) is completed.
- (3) When the power OFF function is called.

8.3.2. Suspend Transition Prohibited Events

The following items are suspend transition prohibited events.

- Key press (other than power key) authentication
- Touch panel tap authentication
- When ActiveSync connection established (IrDA and USB)
- When auto power OFF time is set to "0"
- When the following registry value is set to "0" with a wireless connection established
[HKEY_LOCAL_MACHINE\Comm\CXPort]
"NoldleTimerReset"=dword : 0

Furthermore, the auto power OFF time is reset upon the occurrence of the following events.

- When a serial communication event occurs (IrDA and USB)
- When the SystemIdleTimerReset() function is executed
- When an event with event object name "PowerManager, ActivityTimer, or UserActivity" is set

8.3.3. Setting the Auto Power-off Timeout

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the auto power-off timeout as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Auto power-off timeout (sec.) (When battery-driven)	DW	R/W	BHT_PM_BATTPOWEROFF	0: Disable 1 - 0xFFFFFFFF	180 (3 min.)	Immediately after setting
Auto power-off timeout (sec.) (When placed on the CU)	DW	R/W	BHT_PM_EXTPOWEROFF	0: Disable 1 - 0xFFFFFFFF	0	Immediately after setting

8.3.4. Setting the Effective Held-down Time of the Power Key for Switching to the Suspend State

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the effective held-down time of the power key for switching to the suspend state as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Effective held-down time of the power key for switching to the suspend state (in units of 100 msec)	DW	R/W	BHT_PWRDOWN_KEY_WAIT_TIME	1 - 255	5	Immediately after setting

Saving the Registry

If the BHT is switched to the suspend state by pressing the power key with the SF (*1) key held down, the Registry will be saved into the flash memory.

(*1) Here, this means only the key marked "SF." The Registry will not be saved even if you press the power key while holding down the magic key to which the SF key function is assigned.

Chapter 9. Battery State

9.1. Outline

The battery status can be ascertained using the following methods.





- (1) Battery status acquisition
- (2) Battery voltage icon
- (3) Low battery voltage warning message display

9.2. Battery Voltage Acquisition

The **BHT_GetPowerStatus** function can be used to ascertain whether the BHT is on the CU, and acquires the battery level, battery voltage, and battery type.

9.3. Battery Voltage Icon

The battery voltage status is indicated with the icons below if the battery voltage status display is authorized.

Battery voltage level		Battery Voltage Icon
Level	Voltage	
High	3.9 V or higher	
Medium	3.7 V or higher and less than 3.9 V	
Low	3.6 V or higher and less than 3.7 V	
Warning	3.5V or higher and less than 3.6 V	

9.4. Battery Voltage Warning

If the output voltage of the battery cartridge drops below the specified lower limit, the BHT displays the Level-1 message "Battery voltage has lowered." on the LCD and beeps three times. After that, it will resume the previous regular operation.

If the battery output voltage drops further, the BHT displays the Level-2 message "Charge the battery!," beeps five times, and then turns itself off automatically.

Chapter 10. LED

10.1. Outline

The BHT-700 has two LEDs. The display LED can be controlled from the application.

LED	Color	ON/OFF control from applications
Indicator LED	Red and blue	Possible
Charger LED	Red and green	Impossible

10.2. LED Control

10.2.1. Display LED

(1) Control method

The red and blue display LEDs can be turned ON and OFF using the **BHT_SetNLedStatus**, **BHT_SetNLedOn**, and **BHT_SetNLedOff** functions.

Furthermore, the LED ON/OFF status can be acquired using the **BHT_GetNLedStatus** and **BHT_GetNLedStatusEx** functions

(2) Limited items

- LEDs cannot be controlled when a barcode device file is open. LEDs can be controlled, however, if LEDs are set not to illuminate when a barcode device file is open.
- If the function mentioned at (1) above is used to turn ON an LED from the application, the LED remains ON even after exiting the application used to turn ON the LED. Use the function mentioned at (1) to turn OFF the LED.

10.2.2. Charge LED

The charge LED cannot be turned ON or OFF from the application.

Chapter 11. Data Communication

11.1. Outline

In wired communication between the BHT and host computer, the following interfaces are available:

- (1) IrDA interface
- (2) Connector interface
- (3) USB client interface
- (4) USB host interface

11.2. Programming for Data Communication

(1) IrDA interface

The IrDA interface is assigned to port 4.

Communications parameter	Effective setting	Default
Transmission speed (bps)	115200, 57600, 38400, 19200, 9600	115200

Parameters other than the transmission speed are fixed (Parity = None, Character length = 8 bits, Stop bit length = 1 bit), since the physical layer of the IrDA interface complies with the IrDA-SIR 1.2.

(2) Connector interface

The Connector interface is assigned to port 1.

RTS and CTS signal lines are not supported.

Communications parameter	Effective setting	Default
Transmission speed (bps)	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, 300	115200
Parity	None, even, or odd	None
Character length	7 or 8 bits	8
Stop bit length	1 or 2 bits	1

(3) USB client interface

The USB client interface is used for ActiveSync connection.

(4) USB host interface

The USB host interface is used for connection with network via CU-714 and Ethernet cable. When BHT-700 is set on CU-714, new connection named "AX-887721" is created in Control Panel.

11.3. Assigning Port Number

From COM1 to COM8 are used by system program. Assign 9 or 0 to COM port number of communication made newly.

11.4. ActiveSync

11.4.1. Establishing an ActiveSync Connection

An ActiveSync connection can be established with either of the following procedures.

- (1) Manual connection via the [BhtShell] - [2:Communication] menu
- (2) Automatic connection by placing the BHT on CU-733 connected to the computer by USB

Communication I/F	Manual Connection	Automatic Connection
USB	●	●
IrDA	●	-
Wireless	●	-
USB-LAN (*1)	●	-

(*1) CU-714 is necessary for USB-LAN communication.

11.4.2. ActiveSync Auto Connection Setting Method

The ActiveSync auto connection function is set and read using the **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions.

Parameter	Type	R/W	Control Code	Parameter Value	Default	Validation Timing
ActiveSync auto connection	DW	R/W	BHT_ACTSYNC_AUTOCNCT	ACTSYNC_AUTOCNCT_DISABLE : Prohibited ACTSYNC_AUTOCNCT_USB : USB only permitted	ACTSYNC_AUTOCNCT_USB	Immediately after setting

Chapter 12. Wireless Communication

12.1. Outline

12.1.1. Spread Spectrum Communications Method

Data communication is performed using TCP/IP protocol via a wireless module. Refer to item 13.2 for details on communication program creation.

12.1.2. Configuration of Spread Spectrum System

The BHT communicates with the host computer via an access point in wireless communication.

For details, refer to the "BHT-700BB/700BWB/700BWBG-CE User's Manual" or "BHT-700QWBG-CE User's Manual."

The table below shows the communications status transition as the state of the spread spectrum communications device built in the BHT-700.

Spread spectrum communications device	Communication
Open (power on)	Impossible
Checking synchronization with access point	Impossible
Synchronization complete	Possible
Roaming	Impossible if the BHT is not synchronized with an access point Possible if synchronization with an access point is kept
End of roaming	Possible
Close (power off)	Impossible

If always being opened, the spread spectrum communications device will consume much power. When the device is not in use, therefore, close it as soon as possible.

However, it will take several seconds to open the spread spectrum communications device and synchronize it with the access point for making communications ready. Frequent opening and closing of the device will require much time, resulting in slow response. Take into account the application purposes of user programs when programming.

When the spread spectrum communications device is synchronized with the access point, the BHT will display a synchronization icon on the LCD as shown below.



12.2. Programming for Wireless Communication

To connect to the wireless communications pathway, specify the following system settings in System Menu or in a user program:

- POWER
- RADIO MODE
- ESSID (Extended Service Set ID)
- ENCRYPTION
- AUTHENTICATION
- EAP TYPE
- KEY(WEP KEY, PRE SHARED KEY)

For the procedure in System Menu, refer to the "BHT-700BB/700BWB/700BWBG-CE User's Manual" or "BHT-700QWBG-CE User's Manual."

If no system settings are made in a user program, those made in System Menu will apply.

The following procedure is used to perform system settings in the user program.

Step 1: Select the profile to be edited.

Call the following function to edit an existing profile.

BHT_RF_IoControl (RF_UPDATE_PROFILE, NULL, 0, NULL, 0, NULL);

Call the following function to edit or create a new profile.

BHT_RF_IoControl (RF_SET_PROFILE, ...);

Please refer to section "13.2.1 Wireless Communication Parameters" for details of the setting method.

Use ESSID and Infrastructure mode to specify the profile.

If no profile corresponding to the specified ESSID and Infrastructure mode combination exists, a new profile will be created.

Step 2: Change parameter 1, parameter 2,, parameter N for the profile selected at Step 1.

Please refer to section "13.2.1 Wireless Communication Parameters".

Step 3: Update the set parameters to the driver.

BHT_RF_IoControl (RF_COMMIT_PROFILE, NULL, 0, NULL, 0, NULL);

Use the highest priority profile from among those created to attempt a connection.

If connection fails, attempt to connect automatically using the highest priority profiles sequentially.

The profile with the highest priority will be the one created last.

Up to a maximum of 16 profiles can be created.

12.2.1. Wireless Communication Parameters

Settable Parameters

The BHT can be used with the following security configurations by setting ZeroConfig.

- PEAP (802.1x)
- EAP-TLS (802.1x)
- PEAP (WPA)
- EAP-TLS (WPA)
- PSK (WPA)
- PEAP (WPA2)
- EAP-TLS (WPA2)
- PSK (WPA2)

Details of the parameters used with the above security configurations are outlined in the table below.

Parameter	Security					
	None	PEAP (802.1x)	EAP-TLS (802.1x)	PEAP (WPA)	EAP-TLS (WPA)	PSK (WPA)
Authentication	OPEN	OPEN	OPEN	WPA	WPA	WPA-PSK
Encryption	Disable WEP (static)	WEP (auto distribution)	WEP (auto distribution)	TKIP	TKIP	TKIP
802.1x	Disable	PEAP	EAP-TLS	PEAP	EAP-TLS	Disable
ESSID	●	●	●	●	●	●
Profile Priority	●	●	●	●	●	●
Pre Shared Key	-	-	-	-	-	●
WEP Key	●	-	-	-	-	-

Parameter	Security		
	PEAP (WPA2)	EAP-TLS (WPA2)	PSK (WPA2)
Authentication	WPA2	WPA2	WPA2-PSK
Encryption	AES	AES	AES
802.1x	PEAP	EAP-TLS	Disable
ESSID	●	●	●
Profile Priority	●	●	●
Pre Shared Key	-	-	●
WEP Key	-	-	-

(●: Setting valid, -: Setting invalid)

- **POWER**

Set the power mode for the wireless module built in the BHT. The following two power modes are available. The default is P_PWRSAVE_PSP.

The set value is validated when the wireless LAN device is opened first after setting.

Power mode	Power consuming state
P_PWRSAVE_CAM	Consumes much power (no power saving effect)
P_PWRSAVE_PSP	Consumes less power (much power saving effect). The BHT may take more time to establish the wireless link or send response messages.

[Ex.] Set the power mode to "Consumes much power"

```
DWORD dwVal = P_PWRSAVE_CAM;
```

```
BHT_RF_SetParamInt (P_INT_POWERSAVE, &dwVal, sizeof(dwVal));
```

- **RADIO MODE**

The standard for the wireless LAN being used can be set. The following 3 patterns are available, with the default set to 802.11b.

The set value is validated when the wireless LAN device is opened first after setting.

Radio mode	State
P_RADIOMODE_11A	Operates on wireless LAN standard 802.11a.
P_RADIOMODE_11B	Operates on wireless LAN standard 802.11b.
P_RADIOMODE_11B P_RADIOMODE_11G	Operates on wireless LAN standard 802.11b and 802.11g.

[Ex.] To set the radio mode to "802.11b/g":

```
DWORD dwVal = P_RADIOMODE_11B | P_RADIOMODE_11G;
```

```
BHT_RF_SetParamInt (P_INT_RADIOMODE, &dwVal, sizeof(dwVal));
```

- **ESSID**

Specify an ID that identifies the wireless network as a character string. The ESSID of the BHT should be the same as the SSID of the access point. If the ESSID is not set correctly, no communication is possible.

[Ex.] Set the "BHT700" to the ESSID (The infrastructure mode is assumed to be an "Infrastructure.")

```
ST_RF_PROFILE_KEY stKey;
```

```
wcscpy(&stKey.szESSID[0], TEXT("BHT700")); // ESSID
```

```
stKey.dwInfraMode = INFRA_INFRASTRUCTURE; // Infrastructure
```

```
BHT_RF_IoControl (RF_SET_PROFILE, (LPVOID)&stKey, sizeof(stKey), NULL, 0, NULL);
```

- **ENCRYPTION**

This is the encryption method setting. A selection can be made from Prohibited, WEP, TKIP and AES.

- **AUTHENTICATION**

This is the authentication method setting. A selection can be made from Open, Shared, WPA, WPA-PSK, WPA2, and WPA2-PSK.

- EAP TYPE

This is the EAP type setting. A selection can be made from Prohibited, PEAP, and TLS.

- KEY (WEP KEY, PRE SHARED KEY)

The encryption key (WEP KEY or PRESHARE KEY) can be set.

[Ex.] Setting to enable WEP. Set the WEP KEY to "01234567890123456789ABCDEF" (128 bit).

```
DWORD dwVal = P_AUTH_OPEN;
BHT_RF_SetParamInt (P_INT_AUTHENTICATE, &dwVal, sizeof(dwVal));
DWORD dwVal = P_ENCRYPT_WEP;
BHT_RF_SetParamInt (P_INT_ENCRYPTION, &dwVal, sizeof(dwVal));
DWORD dwVal = P_8021X_DISABLE;
BHT_RF_SetParamInt (P_INT_8021X, &dwVal, sizeof(dwVal));
BHT_RF_SetParamStr (P_STR_WEPKEY1,
                    TEXT("01234567890123456789ABCDEF"),26);
```

Parameter List

Parameter	Type	R/W	Parameter value	Default
Power mode	DW	R/W	P_PWRSERVE_CAM : High power consumption P_PWRSERVE_PSP : Low power consumption	P_PWRSERVE_PSP
Radio mode	DW	R/W	P_RADIOMODE_11A : 802.11a P_RADIOMODE_11B : 802.11b P_RADIOMODE_11B P_RADIOMODE_11G : 802.11b/g	P_RADIOMODE_11B
Authentication method	DW	R/W	P_AUTH_OPEN : Open P_AUTH_SHARED : Shared P_AUTH_WPA : WPA P_AUTH_WPA2 : WPA2 P_AUTH_WPA2PSK : WPA2 PSK	P_AUTH_OPEN
Encryption	DW	R/W	P_ENCRYPT_DISABLE : Prohibited P_ENCRYPT_WEP : WEP P_ENCRYPT_TKIP : TKIP P_ENCRYPT_AES : AES	P_ENCRYPT_DISABLE
802.1x Encryption (EAP type)	DW	R/W	P_8021X_DISABLE : Prohibited P_8021X_PEAP : PEAP P_8021X_TLS : TLS	P_8021X_DISABLE
Profile priority	DW	R/W	1 (high) to 16 (low)	1
Index Key	DW	R/W	1 to 4	1
WEP Key 1	WCS	W	26-character hexadecimal notation character string (128 bit) 10-character hexadecimal notation character string (40 bit)	TEXT("")
Pre Shared Key	WCS	W	8 to 63-character ASCII character string 64-character hexadecimal notation character string	TEXT("")
Version	WCS	R	-	
MAC address	WCS	R	-	TEXT("00.00.00.00.00.00")

Note that if you use **BHT_RF_GetParamInt** function for getting a value, the value preset by the **BHT_RF_SetParamInt** function will be obtained.

12.2.2. Opening and Closing the Wireless Communications Device

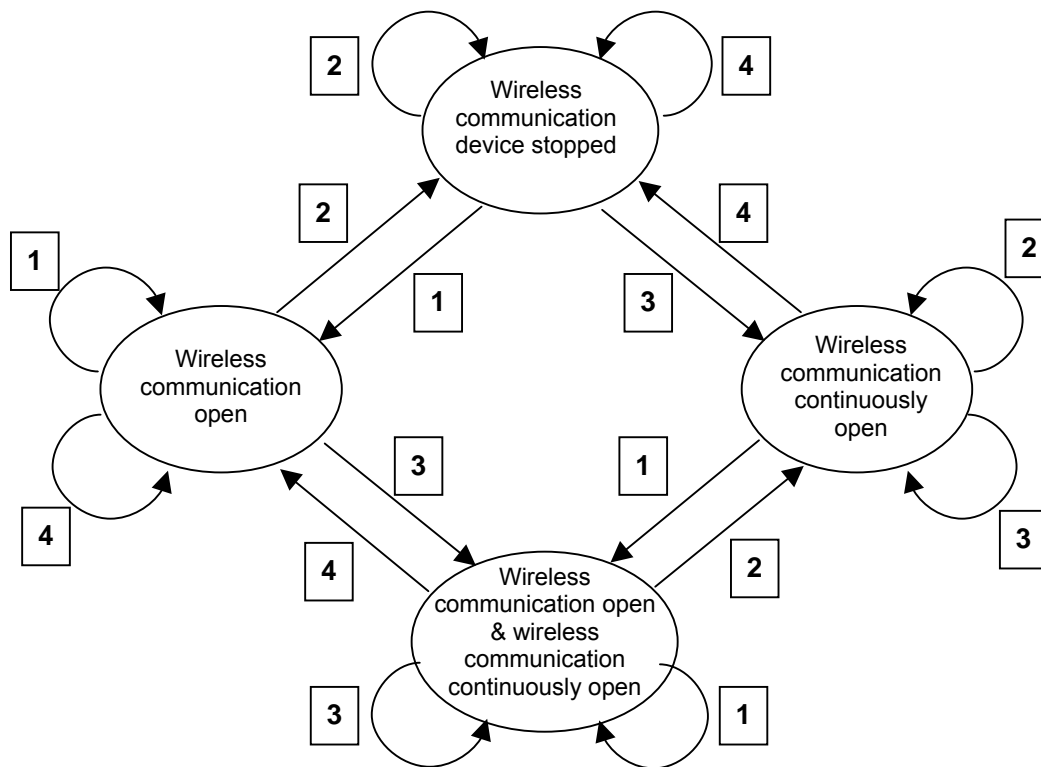
Use the **BHT_RF_Open** and **BHT_RF_OpenEx** functions to start up the wireless communication device and permit wireless communication.

Use the **BHT_RF_Close** and **BHT_RF_CloseEx** functions to stop the wireless communication device and prohibit wireless communication.

Use the **BHT_RF_OpenEx** (DWORD dwOpt) and **BHT_RF_CloseEx** (DWORD dwOpt) functions to perform wireless communication in the following communication formats.

Settable Value	Details
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

The following diagram illustrates the wireless communication device status transmission.



- 1 BHT_RF_Open() (*1)**
- 2 BHT_RF_Close() (*2)**
- 3 BHT_RF_OpenEx(COMM_CONTINUOUS)**
- 4 BHT_RF_CloseEx(COMM_CONTINUOUS)**

(*1) Includes **BHT_RF_OpenEx(COMM_NORMAL)**

(*1) Includes **BHT_RF_OpenEx(COMM_NORMAL)**

12.2.3. Checking Synchronization with the Access Point

When performing data communication with a wireless communication device, use the **BHT_RF_Synchronize** function to check whether synchronization with the access point has been obtained.

The following is a list of possible reasons why it may not be possible to obtain synchronization with the access point.

- (1) The wireless communication device is currently open.
Several seconds are required to obtain synchronization with the access point after opening the wireless communication device.
Furthermore, when using DHCP, there are times when several tens of seconds are required to obtain the IP after connecting to the network.
- (2) When the wireless device is moved from the current access point to the next access point during roaming
- (3) When the wireless device is moved outside the radio-wave area covered by the access point.
- (4) When the wireless device is moved to a location where an obstruction prevents wireless communication with the access point.

Chapter 13. Barcode Reading

13.1. Outline

13.1.1. Enable Reading

BHT-700B

The **BHT_EnableBar** function enables the barcode device to read barcodes. In this function, you may specify the following barcode types available in the BHT. The BHT can handle one of them or their combination.

Available Barcode Type	Default Setting
Universal product codes EAN-13 (*1) (JAN-13 (*1)) EAN-8 (JAN-8) UPC-A (*1), UPC-E	No national flag specified.
Interleaved 2of5 (ITF)	No length of read data specified. No check digit.
Standard 2of5 (STF)	No length of read data specified. No check digit. Short format of the start/stop characters supported.
Codabar (NW-7)	No length of read data specified. No check digit. No start/stop character.
Code 39	No length of read data specified. No check digit.
Code 93	No length of read data specified.
Code 128 (EAN-128) (*2)	No length of read data specified.
Interleaved 2of5 (ITF)	No length of read data specified. No check digit.
RSS	Nothing specified.

(*1) Reading wide bars

EAN-13 and UPC-A barcodes may be wider than the readable area of the barcode reading window. Such wider bars can be read by long-distance scanning. Pull the barcode reading window away from the barcode so that the entire barcode comes into the illumination range.

(*2) Specifying Code 128 makes it possible to read not only Code 128 but also EAN-128.

BHT-700Q

The **BHT_EnableBar** function enables the barcode device to read barcodes. In this function, you may specify the following barcode types available in the BHT. The BHT can handle one of them or their combination.

Available Barcode Type	Default Setting
2D codes	
QR code	Not specified: Model 1, Model 2, Micro QR code, code version No split code scanning
PDF417	PDF417, MicroPDF417
MaxiCode	Nothing specified
Data Matrix	Square Data Matrix, Rectangular Data Matrix Not specified: code no.
EAN·UCC Composite	Nothing specified

1D codes	
EAN-13 (*1) (JAN-13(*1)) EAN-8 (JAN-8) UPC-A *1, UPC-E	No country flag specified No length of read data specified No check digit
Interleaved 2of5 (ITF)	No length of read data specified No check digit No start/stop character
CODABAR (NW-7)	No length of read data specified No check digit No start/stop character
CODE-39	No length of read data specified No check digit
CODE-93	No length of read data specified
CODE-128 (EAN-128)(*2)	No length of read data specified
RSS	Nothing specified

(*1) Reading wide bars

EAN-13 and UPC-A barcodes may be wider than the readable area of the barcode reading window. Such wider bars can be read by long-distance scanning. Pull the barcode reading window away from the barcode so that the entire barcode comes into the illumination range.

(*2) Specifying Code 128 makes it possible to read not only Code 128 but also EAN-128.

13.1.2. Specify Options in the **BHT_EnableBar** Function

You may also specify several options as listed below for each of the barcode types in the **BHT_EnableBar** function.

BHT-700B

Barcode type	Options
Universal product code	Initial (country flag) add-on code
Interleaved 2of5 (ITF)	Length of read data Check digit
CODABAR (NW-7)	Length of read data Start/stop character Check digit
Code 39	Length of read data Check digit
Code 93	Length of read data
Code 128	Length of read data
Standard 2of5(STF)	Length of read data Start/stop character Check digit
MSI	1-digit check digit 2-digit check digit
RSS	Nothing specified.

BHT-700Q

Barcode type	Options
2D codes	
QR	Model Code version Split code scanning
PDF417	Code
MaxiCode	Nothing specified
Data Matrix	Code Code no.
EAN·UCC Composite	Nothing specified
1D codes	
Universal product code	Initial (country flag) add-on code
Interleaved 2of5 (ITF)	Length of read data Check digit
CODABAR (NW-7)	Length of read data Start/stop character Check digit
Code 39	Length of read data Check digit
Code 93	Length of read data
Code 128	Length of read data
RSS	Nothing specified

13.1.3. Barcode Buffer

The barcode buffer stores the inputted barcode data.

BHT-700B

The barcode buffer will be occupied by one operator entry job and can contain up to 99 characters.

BHT-700Q

The barcode buffer will be occupied by one operator entry job and can contain up to 99 bytes in barcode or 8,192 bytes in 2D code (1 kanji character equals 2 bytes).

You can check whether the barcode buffer stores code data, by using the **BHT_GetBarNum** function. To read barcode data stored in the barcode buffer, use the **BHT_ReadBar/BHT_ReadBarEx** function.

13.2. Programming

13.2.1. Code Mark

The **BHT_GetBarType** function allows you to check the code mark (denoting the code type) and the length of the inputted barcode data.

13.2.2. Multiple Code Reading

You may activate the multiple code reading feature which reads more than one code type while automatically identifying them. To do it, you should designate desired code types in the read code parameter of the **BHT_EnableBar** function.

13.2.3. Read Mode of the Trigger Switch

The trigger switch function is assigned to the [SCAN] key and the magic key M3 by default. You may assign the trigger switch function to other keys by using the **BHT_SysSettingDW** function. You may select the read mode of the trigger switch by using the **BHT_EnableBar** function as listed below.

Read Mode	BHT_EnableBar Function
Auto-off Mode	BHT_EnableBar (TEXT ("F...
Momentary Switching Mode	BHT_EnableBar (TEXT ("M...
Alternate Switching Mode	BHT_EnableBar (TEXT ("A...
Continuous Reading Mode	BHT_EnableBar (TEXT ("C...

To check whether the trigger switch is pressed or not, use the **BHT_WaitEvent** function as shown below.

```
BHT_WaitEvent (1, BHT_EVT_MASK_TRGDOWN, 0, &dwSignaledEvent);  
if ( (dwSignaledEvent & BHT_EVT_MASK_TRGDOWN) != 0 ) {  
    printf("Trigger switch pressed ");  
}
```

13.2.4. Generating a Check Digit of Barcode Data

Specifying a check digit in the **BHT_EnableBar** function makes the Interpreter automatically check barcodes. If necessary, you may use the **BHT_GetBarChkdgt** function for generating a check digit of barcode data.

13.2.5. Controlling the Indicator LED and Beeper/Vibrator as a Confirmation of Successful Reading

By using the **BHT_EnableBar** function, you can control:

- whether the indicator LED should light in blue or not (Default: Light in blue)
- whether the beeper should beep or not (Default: No beep)

when a barcode is read successfully. For detailed specifications, refer to the description for the **BHT_EnableBar** function.

It is also possible to operate the vibrator as a confirmation of successful reading instead, by using the **BHT_SetSysSettingDW** (BHT_BEEP_VIB_SELECT, VIB_SELECT) function.

(1) Controlling the indicator LED

If you have activated the indicator LED (blue) in the **BHT_EnableBar** function, an application cannot control the LED.

If you have deactivated the indicator LED (blue) in the **BHT_EnableBar** function, an application can control the LED even when the barcode device file is opened.

This way, you can control the indicator LED, enabling that:

- a user program can check the value of a scanned barcode and turn on the indicator LED in blue when the barcode has been read successfully.
- a user program can turn on the indicator LED in red the moment the barcode has been read.

(2) Controlling the beeper and vibrator

If you have activated the beeper in the **BHT_EnableBar** function, the BHT will beep when it reads a barcode successfully.

You may select beeping only, vibrating only, or beeping & vibrating by setting on the system menu (BhtShell.exe) or by setting the output port in the **BHT_SetSysSettingDW**(BHT_BEEP_VIB_SELECT,...).

This feature is used to sound the beeper or operate the vibrator the moment the BHT reads a barcode successfully.

13.2.6. Reading Split QR Codes (Only for BHT-700Q)

The QR Code symbology can split data into a maximum of 16 blocks and encodes each of them into a split code image. When those split code images are scanned, the splitter system restores them into the original data string in any of the following three modes--edit mode, batch edit mode, and non-edit mode. These modes can be specified by **BHT_EnableBar** as follows:

Split code scanning mode	BHT_EnableBar function
Edit mode	BHT_EnableBar (..., TEXT("Q : E"))
Batch edit mode	BHT_EnableBar (..., TEXT("Q : B"))
Non-edit mode	BHT_EnableBar (..., TEXT("Q : C"))

In edit mode, after completion of reading all split code images, the splitter system stores the read data into the code buffer. In batch edit mode, when all split code images that fall within the scanning range are read, the splitter system stores the read data into the code buffer. In non-edit mode, each time a single split code image is read, the splitter system stores the read data into the code buffer.

The code type which is acquired by the **BHT_GetBarType** function and the **BHT_GetBarInfo** function is "Q" in edit mode and batch edit mode or "S" in non-edit mode.

NOTE: In the Point Scan mode, scanning split codes in batch edit mode is disabled. (For details about the Point Scan mode, refer to the "BHT-700BB/700BWB/700BWBG-CE User's Manual" or "BHT-700QWBG-CE User's Manual").

13.3. Barcode Reading Using the Virtual COM Port

13.3.1. Outline

Barcode reading using the virtual COM port is supported on the BHT-700 series.

For greater convenience, this function is available for use in conjunction with kbifCE. For more information on kbifCE, see the kbifCE user's guide (available for download on the DENSOWAVE QBNet website).

Using this function it is possible to obtain reading data as if it were being received through a COM port. For applications, it is equivalent to a reader being connected to the communication port (COMx). Using COM, barcode reading data can be used by multiple applications.

13.3.2. Programming

Port number 5 is allocated to the virtual COM port used for barcode reading.

Barcode reading mode and the types of barcodes that are allowed to be read are designated by the kbifCE.

A comparison of the functions of Win32 API when using a general COM port and a virtual COM port for barcode use is as follows:

Win32 API	General COM	Virtual COM used for reading
CreateFile	Open COM port	←
CloseHandle	Close COM port	←
ReadFile	Read received data	Read data
GetCommMask	Obtain type of wait event	←
SetCommMask	Set type of wait event	← Treat completed reading event as receiving event. Non-reading events invalid.(*1)
GetCommTimeouts	Obtain timeout value	←
SetCommTimeouts	Set timeout value	← Non-receiver side timeouts invalid.(*1)
WaitCommEvent	Wait for event	← Non-receiving events invalid.

(*1) An error will not occur.

The following functions are not supported. If operation is attempted, no function will be executed.

List of functions not yet supported		
WriteFile	GetCommModemStatus	SetCommBreak
ClearCommBreak	GetCommProperties	SetCommState
ClearCommError	GetCommState	SetupComm
EscapeCommFunction	PurgeComm	TransmitComm

13.3.3. How to Use

Start up kbifCE and set the destination for the virtual COM port (for further details see the kbifCE user's guide).

Chapter 14. Updating OS

The OS can be updated (version update) using the following method when running Windows CE.

(1) Copy the OS image file to an arbitrary folder.

(2) Execute the **BHT_SystemModify** function.

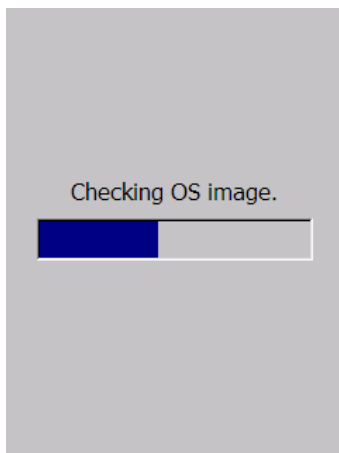
For the 1st argument, specify the absolute path to the folder in which the OS image file was stored, and for the 2nd argument, specify whether to turn OFF the power or perform a cold boot after updating the OS.

(For example) Update OS image file named “_B7BWDW0.SY3” stored in “temp\” folder and perform a cold boot after updating OS

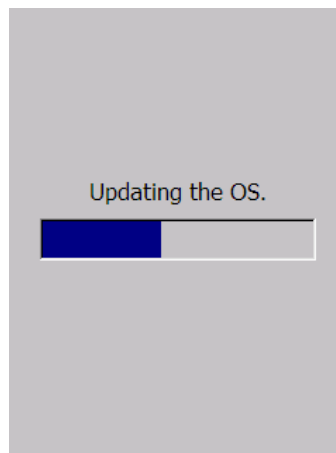
```
DWORD dwRtn;
```

```
dwRtn = BHT_SystemModify(TEXT(\\Temp\\_B7BWDW0.SY3), SYSMDFY_REBOOT);
```

(3) Following display is shown during updating OS.



(checking OS image)



(writing OS image)

(4) After the OS has been successfully updated, the BHT-700 power will either be turned OFF or will cold boot depending on the setting made for the 2nd argument.

Chapter 15. System Functions

System functions are used when setting or acquiring system values or when acquiring device information.

Function	Used to:
BHT_SetSysSettingDW	Write system parameter values (DWORD).
BHT_GetSysSettingDW	Read system parameter values (DWORD).
BHT_SetSysSettingWCS	Write system parameter values (character string).
BHT_GetSysSettingWCS	Read system parameter values (character string).
BHT_GetDeviceInfo	Device information acquisition

15.1.If a System Parameter Value is DWORD

BHT_SetSysSettingDW

Description

Write system parameter values.

Syntax

```
DWORD BHT_SetSysSettingDW (  
    DWORD dwCtrlCode ,  
    DWORD dwSysParam )
```

Parameters

dwCtrlCode

[in] Control code

dwSysParam

[in] Parameter value

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid parameter
ERROR_GEN_FAILURE	Not supported

BHT_GetSysSettingDW

Description

Read system parameter values.

Syntax

```
DWORD BHT_GetSysSettingDW (  
    DWORD dwCtrlCode ,  
    DWORD* pdwSysParam )
```

Parameters

dwCtrlCode

[in] Control code

pdwSysParam

[out] Address for storing the parameter value

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_GEN_FAILURE	Not supported
ERROR_INVALID_PARAMETER	No storage address specified

15.2. If a System Parameter Value is a Character String

BHT_SetSysSettingWCS

Description

Write system parameter values.

Syntax

```
DWORD BHT_SetSysSettingWCS (  
  DWORD dwCtrlCode ,  
  TCHAR* pwchSysParam ,  
  DWORD dwLen )
```

Parameters

dwCtrlCode

[in] Control code

pwchSysParam

[in] Heading address of the storage buffer for a string written

dwLen

[in] String length

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid parameter
ERROR_GEN_FAILURE	Not supported

BHT_GetSysSettingWCS

Description

Read system parameter values.

Syntax

```
DWORD BHT_GetSysSettingWCS (  
    DWORD dwCtrlCode ,  
    TCHAR* pwchSysParam ,  
    DWORD dwLen ,  
    DWORD* pdwLenReturned )
```

Parameters

dwCtrlCode

[in] Control code

pwchSysParam

[out] Heading address of the storage buffer for a string read

dwLen

[in] String length

pdwLenReturned

[out] Length of the string read out

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_GEN_FAILURE	Not supported
ERROR_INVALID_PARAMETER	No storage address specified

15.3. System Parameter Values That Can be Set/Obtained

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
System information related						
System version (4 characters)	WCS	R	BHT_SYS_OS_VERSION	-	-	-
Total RAM size (bytes)(*1)	DW	R	BHT_SYS_RAMSIZE	-	-	-
Total ROM size (bytes) (*1)	DW	R	BHT_SYS_ROMSIZE	-	-	-
Model name (8 characters)	WCS	R	BHT_SYS_MACHINE_NAME	-	-	-
Product number (16 characters)	WCS	R	BHT_SYS_MACHINE_NUMBER	-	-	-
Serial number (6 characters)	WCS	R/W	BHT_SYS_SERIAL_NUMBER	6-digit number	Lower 6 characters in the code printed on the back of the BHT	Immediately after setting
Power management related						
Waiting time to switch to standby mode (in units of 100 ms)	DW	R/W	BHT_PM_STBYTIME	0: Disable 1 to 255	10 (1 sec)	Immediately after setting
Waiting time to auto power OFF when powered by battery (sec.)	DW	R/W	BHT_PM_BATTPOWEROFF	0: Disable 1 to 0xFFFFFFFF	180 (3 min)	Immediately after setting
Waiting time to auto power OFF when placed on CU (sec.)	DW	R/W	BHT_PM_EXTPOWEROFF	0: Disable 1 to 0xFFFFFFFF	0	Immediately after setting
CPU clock (*2)	DW	R/W	BHT_PM_CPU_CLOCK	CPU_CLK_NORMAL : Regular speed CPU_CLK_FAST : High speed	CPU_CLK_NORMAL	When warm-booting after setting
Auto Power OFF permitted when wireless connection open	DW	R/W	BHT_PM_SUSPEND_RF	SUSPEND_DISABLE : Suspend prohibited SUSPEND_ENABLE : Suspend permitted	SUSPEND_ENABLE	Immediately after setting
Beeper and vibrator related						
Rumble device	DW	R/W	BHT_BEEP_VIB_SELECT	BEEP_SELECT : Beeper VIB_SELECT : Vibrator (BEEP_SELECT VIB_SELECT) : Beeper and vibrator	BEEP_SELECT	first sound after setting
Beeper volume	DW	R/W	BHT_BEEP_VIB_VOLUME	0: OFF 1 (lowest) to 5 (highest)	5	first sound after setting
Key click volume	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1: Soft 2: Loud	2	first sound after setting
Screen tap volume	DW	R/W	BHT_BEEP_VIB_TAP	0: OFF 1: Soft 2: Loud	2	first sound after setting
Trigger switch clicks(*3)	DW	R/W	BHT_BEEP_VIB_TRGKEY	CLICK_SOUND_OFF : Prohibit CLICK_SOUND_ON : Allow	CLICK_SOUND_OFF	first sound after setting

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Backlight related						
Backlight ON-duration (sec.) (When battery- driven)	DW	R/W	BHT_BACKLIGHT_BATT_TIME	0 - 255 0: Backlight OFF 255: Backlight continuously ON	3	When backlight illumination timer is next reset
Backlight ON-duration (sec.) (When placed on the CU)	DW	R/W	BHT_BACKLIGHT_AC_TIME	0 - 255 0: Backlight OFF 255: Backlight continuously ON	60	When backlight illumination timer is next reset
Control key	DW	R/W	BHT_BACKLIGHT_KEY	Key number	0x10203 ([SF]+[M3])	Immediately after setting
Backlight brightness level	DW	R/W	BHT_BACKLIGHT_BRIGHTNESS	0: OFF 1: Dark – 3: Bright	3	When the backlight is next turned ON
Backlight power saving mode	DW	R/W	BHT_BACKLIGHT_POWERSAVE	0: OFF 1: Dim	1	When backlight illumination status is set to power saving mode first after setting
Illumination device selection when backlight illumination specified from API	DW	R/W	BHT_BACKLIGHT_DEVICE	0: None LIGHTING_LCD :LCD LIGHTING_KEY :KEY LIGHTING_LCD LIGHTING_KEY : Both LCD and key	LIGHTING_LCD	Immediately after setting, when BHT_SetBitStatus next called
Key backlight illumination trigger	DW	R/W	BHT_BACKLIGHT_FACTOR	0: Always OFF BHT_BLT_KEY_FACTOR_KEY : Key-press BHT_BLT_KEY_FACTOR_KEYTAP : Both key-press and tap	BHT_BLT_KEY_FACTOR_KEY	Immediately after setting, first tap or key press when "BHT_BLT_KEY_FACTOR_KEY" or "BHT_BLT_KEY_FACTOR_KEYTAP"
Barcode reading related						
Re-read prevention enabled time (in units of 100 ms)	DW	R/W	BHT_BAR_CRTIME	0 to 255 (*4)	10	Immediately after setting
Black-and-white inverted label reading function	DW	R/W	BHT_BAR_INVERT	<u>BHT-700B</u> 0: Prohibit 1: Allow (automatic) <u>BHT-700Q</u> 0: Disable 1. Enable (black-and-white inversion only) 2: Allow (automatic)	0	Immediately after setting
Decode level	DW	R/W	BHT_BAR_DCD_LEVEL	1 to 9	4	When the barcode device is opened first after setting
Min. number of digits to be read for ITF	DW	R/W	BHT_BAR_MINDGT_ITF	2 to 20	4	When the barcode device is opened first after setting
Min. number of digits to be read for STF (*5)	DW	R/W	BHT_BAR_MINDGT_STF	1 to 20	3	When the barcode device is opened first after setting
Min. number of digits to be read for Codabar (CODABAR)	DW	R/W	BHT_BAR_MINDGT_NW7	3 to 20	4	When the barcode device is opened first after setting

Scanning range marker (*6)	DW	R/W	BHT_BAR_MARKER	MARKER_NORMAL : Normal mode MARKER_AHEAD : Maker ahead MARKER_DISABLE : Fixed to OFF	MARKER _NORMAL	Immediately after setting
-------------------------------	----	-----	----------------	-----------------------------------------------------------------------------------------------------	-------------------	------------------------------

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Front-back inverted reading (*6)	DW	R/W	BHT_BAR_REVERSE	0: Disable 1: Enable	0	Immediately after setting
Scan mode (*6)	DW	R/W	BHT_BAR_SCAN_MODE	SCAN_MODE_NORMAL : Normal mode SCAN_MODE_POINT : Point scan mode SCAN_MODE_1D :Barcode reader mode	SCAN_MODE_NORMAL	When the barcode device is opened first after setting
Option data (*6)	DW	R/W	BHT_BAR_OPTION_DATA	0: There is option data. 1: No option data	0	Immediately after setting
Illumination mode (*6)	DW	R/W	BHT_BAR_LIGHT_MODE	0:AUTO 1: Always ON 2: Always OFF	1	Immediately after setting
Keyboard related						
Shift key mode	DW	R/W	BHT_KEY_SHIFT_MODE	KEY_NON_LOCK : Non-lock KEY_ONE_TIME : Onetime lock	KEY_NON_LOCK	Immediately after setting
Assignment to M1 key	DW	R/W	BHT_KEY_M1_MODE	MAGIC_FUNC_NONE : Ignore the depressed key MAGIC_FUNC_ENTER : Treat as ENT key MAGIC_FUNC_TRG : Treat as trigger switch MAGIC_FUNC_SHIFT : Treat as SF key	MAGIC_FUNC_TAB	Immediately after setting
Assignment to M2 key	DW	R/W	BHT_KEY_M2_MODE	MAGIC_FUNC_ALT : Treat as ALT key MAGIC_FUNC_CTRL : Treat as CTRL key MAGIC_FUNC_BLT : Treat as bacjlight function on/off key	MAGIC_FUNC_NONE	Immediately after setting
Assignment to M3 key	DW	R/W	BHT_KEY_M3_MODE	MAGIC_FUNC_TAB : Treat as TAB key MAGIC_FUNC_CLEAR : Treat as CLEAR key MAGIC_FUNC_USERDEF : User-defined code	MAGIC_FUNC_TRG	Immediately after setting
Key entry mode	DW	R/W	BHT_KEY_INPUT_METHOD	INPUT_METHOD_NUMERIC : Numeric entry mode INPUT_METHOD_ALPHABET2 : Alphabet entry mode 2	27-key type: INPUT_METHOD_NUMERIC 42-key type: INPUT_METHOD_ALPHABET	Immediately after setting
Enable/disable alphabet entry switching key	DW	R/W	BHT_DISABLE_KEYMODE_CHANGE_KEY	ENABLE_KEY_TOCHANGE_ALPHABET : Enable DISABLE_KEY_TOCHANGE_ALPHABET : Disable	ENABLE_KEY_TOCHANGE_ALPHABET	Immediately after setting
Function mode	DW	R/W	BHT_KEY_FUNCTION	KEY_FUNCTION_ON : Function mode KEY_FUNCTION_OFF : Non-function mode	KEY_FUNCTION_OFF	Immediately after setting
Effective held-down time of power key for suspending (in units of 100 ms)	DW	R/W	BHT_PWRDOWN_KEY_WAIT_TIME	1 - 255	5	Immediately after setting
Keypad type	DW	R	BHT_KEYBOARD_TYPE	KEYBOARD_TYPE1 : 27-key pad KEYBOARD_TYPE2 / KEYBOARD_TYPE2P : 42-key pad	-	-

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Status indicator related						
Battery voltage level icon	DW	R/W	BHT_ICON_BATTERY	0: Hide 1: Display	1	Immediately after setting
Software keyboard icon	DW	R/W	BHT_ICON_SIP	0: Hide 1: Display	1	Immediately after setting
Keypad shift icon	DW	R/W	BHT_ICON_SHIFTKEY	0: Hide 1: Display	1	The icon appears when the keypad is shifted first after this parameter is set to "1." (If the keypad has been shifted, the icon appears immediately.) It disappears when the shift is released first after this parameter is set to "0."
Alphabet input icon (27-key type only)	DW	R/W	BHT_ICON_IN_ALPHA	0: Hide 1: Display	1	The icon appears when the alphabet input function is activated first after this parameter is set to "1." It disappears when the alphabet input function is deactivated first after this parameter is set to "0."
Wireless communication state icon	DW	R/W	BHT_ICON_RADIO_INTENSE	0: Hide 1: Display	1	The icon appears when the wireless device is opened first after this parameter is set to "1." (If the wireless device has been opened, the icon appears immediately.) It disappears immediately after this parameter is set to "0."
Standby state icon	DW	R/W	BHT_ICON_STANDBY	0: Hide 1: Display	0	The icon appears when the CPU comes to be on standby first after this parameter is set to "1." It disappears immediately after this parameter is set to "0."
Function mode state icon	DW	R/W	BHT_ICON_FUNCTION	0: Hide 1: Display	1	The icon appears when the function mode is activated first after this parameter is set to "1." It disappears when the function mode is deactivated first after this parameter is set to "0."
Numeric entry mode icon (42-key type only)	DW	R/W	BHT_ICON_NUMERIC	0: Hide 1: Display	1	The icon appears when the numeric entry mode is activated first after this parameter is set to "1." It disappears when the numeric entry mode is deactivated first after this parameter is set to "0."
Bluetooth power status	DW	R/W	BHT_ICON_BLUETOOTH	0: Hide 1: Display	0	Immediately after setting
Communication related						
ActiveSync automatic connection	DW	R/W	BHT_ACTSYNC_AUTOCNCT	ACTSYNC_AUTOCNCT_DISABLE : Prohibited ACTSYNC_AUTOCNCT_USB : Only USB allowed	ACTSYNC_AUTOCNCT_USB	After setting, when the USB cable or RS232C cable is first inserted, or when the CU421 is installed.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Display related						
Screen rotation	DW	R/W	BHT_DISP_ROTATION	DISP_ROTATION_0 : 0° DISP_ROTATION_90 : 90° DISP_ROTATION_180 : 180° DISP_ROTATION_270 : 270°	DISP_ROTATION_0	Immediately after setting
Screen rotation control key	DW	R/W	BHT_DISP_ROTATION_KEY	Key no.	None	Immediately after setting
Touch screen related						
Touch screen disabling	DW	R/W	BHT_TOUCH_DEVICE	TOUCH_ENABLE : Enable TOUCH_DISABLE : Disable	TOUCH_ENABLE	Immediately after setting
Bluetooth related						
Bluetooth device initial power status	DW	R/W	BHT_BT_INITIAL_POWER_STATUS	BHT_BT_POWER_OFF : Power OFF BHT_BT_POWER_ON : Power ON	BHT_BT_POWER_OFF	Immediately after setting
Audio related						
Voice output from receiver	DW	R/W	BHT_AUDIO_OUT_RCV	BHT_AUDIO_OUT_RCV_DISABLE : Disable BHT_AUDIO_OUT_RCV_ENABLE : Enable	BHT_AUDIO_OUT_RCV_DISABLE	Immediately after setting

- (*1) The RAM or ROM size obtained indicates the capacity of the memory mounted on the BHT. To obtain the size of the memory area allowed for the user to use, use **GetDiskFreeSpaceEx**.
- (*2) If the CPU clock is set to high speed, the processing speed becomes higher but the power consumption increases.
- (*3) This parameter controls the on/off of the click sound of the magic key which the trigger switch is assigned to. If it is set to ON, pressing the magic key clicks at the volume specified by the "Key clock volume."
- (*4) If this parameter is set to "0," the BHT no longer reads the same barcode in succession.
- (*5) Only for BHT-700B
- (*6) Only for BHT-700Q

15.4. Device Information Acquisition

BHT_GetDeviceInfo

Description

Acquires the mounted device information.

Syntax

```
DWORD BHT_GetDeviceInfo(  
    DWORD dwDevice,  
    DWORD* pdwDevInfo,  
    LPVOID pvExtInfo )
```

Parameters

dwDevice

[in] Type of the device for which information is being acquired

pdwDevInfo

[out] Device information storage location

pdwDevInfo

[out] Extension information storage location

Return Value

Error Code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error

Device Type	dwDevice	*pdwDevInfo
Read device	BHT_DEV_SCANNER	Either of the following : SCANNER_1DIM :1D SCANNER_2DIM :2D
Communication device	BHT_DEV_COM	The following combinations: COMDEV_SERIAL :RS-232C COMDEV_INFRARED :Infrared Port COMDEV_USB :USB COMDEV_RF :Wireless COMDEV_BLUETOOTH :Bluetooth COMDEV_GPRS :GPRS
ActiveSync device	BHT_DEV_ACTSYNC	The following combinations: ACTSYNC_SERIAL :RS-232C ACTSYNC_INFRARED :Infrared Port ACTSYNC_USB :USB ACTSYNC_RF :Wireless

Chapter 16. Device Control Functions

The device control functions listed below control the devices (barcode reading device, backlight, battery, indicator LED, etc.) dedicated to the BHT.

Function	Used to:
BHT_EnableBar	Open the barcode device file to enable barcode reading. This function specifies the read mode and readable barcode types.
BHT_DisableBar	Close the barcode device file to disable barcode reading.
BHT_ReadBar	Read out data read from the barcode buffer.
BHT_ReadBarEx	Read out data from the barcode buffer and encodes it into the specified data format.
BHT_GetBarType	Read the barcode type and the number of digits of a barcode read most recently.
BHT_GetBarNum	Read the number of digits of the barcode remaining in the barcode buffer.
BHT_GetBarInfo	Read the information on the code read most recently.
BHT_GetBarChkDgt	Calculate a check digit (CD) of the barcode data according to the calculation method specified by dwCDType.
BHT_BAR_SetDecodeOptions	Sets the editing function setting value for the decoded result.
BHT_BAR_GetDecodeOptions	Acquires the editing function setting value for the decoded result.
BHT_SetBlitStatus	Control the backlight.
BHT_GetBlitStatus	Read the backlight status.
BHT_GetPowerStatus	Read information about the battery loaded in the BHT body.
BHT_GetPowerStatus2nd	Read information about the battery loaded in the grip.
BHT_GetNLedStatus	Read the status of the indicator LED.
BHT_SetNLedStatus	Control the indicator LED.
BHT_GetNLedStatusEx	Read the status of the indicator LED and synchronization LED.
BHT_SetNLedOn	Turn on the indicator LED and/or synchronization LED.
BHT_SetNLedOff	Turn off the indicator LED and/or synchronization LED.
BHT_StartBeep	Drive the beeper/vibrator.
BHT_StartBeeperOnly	Drive the beeper.
BHT_StartVibrationOnly	Drive the vibrator.

Function	Used to:
BHT_RF_Open	Open the wireless LAN device and enable wireless communication.
BHT_RF_OpenEx	Set the communication format, open the wireless LAN device and enable wireless communication.
BHT_RF_Close	Close the wireless LAN device and disable wireless communication.
BHT_RF_CloseEx	Close the wireless LAN device for the set format and disable wireless communication.
BHT_RF_Synchronize	Get the association status.
BHT_RF_GetParamInt	Read integer from the wireless communications parameter.
BHT_RF_SetParamInt	Write integer to the wireless communications parameter.
BHT_RF_GetParamStr	Read string from the wireless communications parameter.
BHT_RF_SetParamStr	Write string to the wireless communications parameter.
BHT_RF_GetInfoInt	Read integer from the communications parameter.
BHT_RF_GetInfoStr	Read string to the communications parameter.
BHT_RF_IoControl	Perform operation for the profile and certificate etc.
BHT_RF_GetSiteSurvey	Get quality of the communications link.
BHT_SystemModify	Update the BHT OS.
BHT_WaitEvent	Make the system wait until the specified event or timeout occurs.
BHT_WaitStandbyEvent	Make the system wait until the specified event occurs.
BHT_ShutdownSystem	Turn off the BHT and boot it according to the specified mode.
BHT_RegStore	Turn off the BHT and boot it according to the specified mode.

16.1. Barcode API

BHT_EnableBar

Description

Open the barcode device file to enable barcode reading.

This function specifies the read mode and readable barcode types. Up to eight barcode types can be specified.

Syntax

```
DWORD BHT_EnableBar (  
TCHAR* pwchRdMode ,  
TCHAR* pwchCdParam )
```

Parameters

pwchRdMode

[in] Heading address of the storage buffer for a character string specifying the read mode, beeper/vibrator on/off, and LED on/off

pwchCdParam

[in] Heading address of the storage buffer for a character string specifying barcode types to be read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_TOO_MANY_OPEN_FILES	Barcode device file already opened.
ERROR_INVALID_PARAMETER	Parameter error. More than 24 barcode types are specified.

Comment:

Up to 24 barcode types can be specified.

BHT-700Q:

The maximum code version for QR Code, the maximum code number for Data Matrix, and the maximum number of digits for barcodes are limited by the readable range.

■ Readmode

The BHT supports four read modes--the momentary switching mode, the auto-off mode, the alternate switching mode, and the continuous reading mode, which can be selected by specifying M, F, A, and C to readmode, respectively.

□ Momentary switching mode (M)

Only when you hold down the trigger switch, the illumination LED lights and the BHT can read a barcode.

Until the entered barcode data is read out from the barcode buffer, pressing the trigger switch cannot turn on the illumination LED so that the BHT cannot read the next barcode.

[Ex]

BHT_EnableBar (TEXT ("M"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

□ Auto-off mode (F)

If you press the trigger switch, the illumination LED comes on. When you release the switch or when the BHT completes barcode reading, then the illumination LED will go off. Holding down the trigger switch lights the illumination LED for a maximum of 5 seconds.

While the illumination LED is on, the BHT can read a barcode until a barcode is read successfully or the barcode device file becomes closed.

If the illumination LED goes off after 5 seconds from when you press the trigger switch, it is necessary to press the trigger switch again for reading a barcode.

Once a barcode is read successfully, pressing the trigger switch cannot turn on the illumination LED and the BHT cannot read the next barcode as long as the entered barcode data is not read out from the barcode buffer.

[Ex]

BHT_EnableBar (TEXT ("F"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

□ Alternate switching mode (A)

If you press the trigger switch, the illumination LED comes on. Even if you release the switch, the illumination LED remains on until the barcode device file becomes closed or you press that switch again. While the illumination LED is on, the BHT can read a barcode.

Pressing the trigger switch toggles the illumination LED on and off.

Once a barcode is read successfully, pressing the trigger switch turns on the illumination LED but the BHT cannot read the next barcode as long as the entered barcode data is not read out from the barcode buffer.

[Ex]

BHT_EnableBar (TEXT("A"), TEXT("A,I:4-99,M:1-99,N:3-99,L:1-99,K:1-99,H:1-99"))

□ Continuous reading mode (C)

If this mode is specified, the BHT turns on the illumination LED and keeps it on until the barcode device file becomes closed, irrespective of the trigger switch.

While the illumination LED is on, the BHT can read a barcode.

Once a barcode is read successfully, the BHT cannot read the next barcode as long as the entered barcode data is not read out from the barcode buffer.

[Ex]

BHT_EnableBar (TEXT("C"), TEXT("A,I:4-99,M:1-99,N:3-99,L:1-99,K:1-99,H:1-99"))

In the momentary switching mode, alternate switching mode, or continuous reading mode, after you read a low-quality barcode which needs more than one second to be read, keeping applying the barcode reading window to that barcode may re-read the same barcode in succession at intervals of one second or more.

■ Beepercontrol and LEDcontrol

This function can control the beeper and the indicator LED to activate or deactivate each of them when a barcode is read successfully. This function may also control the vibrator with beepercontrol.

- You should describe parameters of readmode, beepercontrol, and LEDcontrol without any space inbetween.
- You should describe readmode, beepercontrol, and LEDcontrol in this sequence.
- Specifying B to beepercontrol allows you to select beeping only, vibrating only, or beeping & vibrating according to the setting made on the BEEP/VIBRATOR menu in System Menu or the setting made with the system function.
- Specifying L to LEDcontrol will not turn on the indicator LED.

[Ex] To sound the beeper (or operate the vibrator) when a barcode is read successfully:

BHT_EnableBar (TEXT("FB"), TEXT("A,I:4-99,M:1-99,N:3-99, L:1-99,K:1-99,H:1-99"))

[Ex] To deactivate the indicator LED when a barcode is read successfully:

BHT_EnableBar (TEXT ("FL"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

■ Readcode

BHT-700B

The BHT supports the universal product codes, Interleaved 2of5 (ITF), Standard 2of5 (STF), Codabar (NW-7), Code 39, Code 93, and Code 128, MSI, and RSS. The BHT can read also EAN-128 if Code 128 is specified.

- Universal product codes (A)

Syntax

A[:[code]][1st character [2nd character]][supplemental]]

where code is A, B, or C specifying the following:

code	Barcode type
A	EAN-13 (JAN-13), UPC-A
B	EAN-8 (JAN-8)
C	UPC-E

If code is omitted, the default is all of the universal product codes.

1stchara and 2ndchara are flag characters representing a country code and should be numerals from 0 to 9. If a question mark (?) is specified to 1stchara or 2ndchara, it acts as a wild card.

“supplemental” refers to the reading of an add-on code. Specifying an S for add-on enables the BHT to read also barcodes with an add-on code.

[Ex] To enable the BHT to scan EAN-13 with 1stchara "4," 2ndchara "9" and add-on code

BHT_EnableBar(TEXT("FL"), TEXT("A:49S"))

[Ex] To enable the BHT to scan EAN-13 and EAN-8 only

BHT_EnableBar(TEXT("FL"), TEXT("A:A,A:B"))

- Interleaved 2 of 5 (ITF) (I)

Syntax

`I[:[mini.no.digits[-max.no.digits]][CD]]`

where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 2 to 99 and satisfy the following conditions:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BhtShell.exe) up to 99 digits.

If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C to CD makes the Interpreter check barcodes with MOD-10. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan ITF with min.no.digits 6, max.no.digits 10, and MOD-10

BHT_EnableBar(TEXT("FL"), TEXT("I:6-10C"))

[Ex] To enable the BHT to scan ITF with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40

BHT_EnableBar(TEXT("FL"), TEXT("I:6-10,I:20-40"))

□ CODABAR (NW-7) (N)

Syntax

N[:[mini.no.digits[-max.no.digits]][startstop]][CD]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 3 to 99 and satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BhtShell.exe) up to 99 digits.

If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

start and stop are the start and stop characters, respectively. Each of them should be an A, B, C, or D. If a question mark (?) is specified, it acts as a wild card. The start and stop characters are included in the number of digits. The A through D will be stored in the barcode buffer as a through d.

CD is a check digit. Specifying a C to CD makes the Interpreter check barcodes with MOD-16. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan NW-7 with min.no.digits 8, start character A and stop character A, and MOD-16

BHT_EnableBar(TEXT("FL"), TEXT("N:8AAC"))

[Ex] To enable the BHT to scan NW-7 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40

BHT_EnableBar(TEXT("FL"),TEXT("N:6-10,N:20-40"))

□ CODE-39 (M)

Syntax

M[:[mini.no.digits[-max.no.digits]][CD]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C to CD makes the Interpreter check barcodes with MOD-43. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan Code 39 with min.no.digits 8, max.no.digits 12, and MOD-43
BHT_EnableBar(TEXT("FL"), TEXT("M:8-12C"))

[Ex] To enable the BHT to scan Code 39 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("M:6-10,M:20-40"))

□ CODE-93 (L)

Syntax

L[:[mini.no.digits[-max.no.digits]]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 1 to 99, excluding start/stop characters and check digits. They should satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

[Ex] To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 12

BHT_EnableBar(TEXT("FL"), TEXT("L:6-12"))

[Ex] To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40

BHT_EnableBar(TEXT("FL"),TEXT("L:6-10,L:20-40"))

NOTE: Neither start/stop characters nor check digits will be transferred to the barcode buffer.

□ CODE-128 (K)

Syntax

K[:[mini.no.digits[-max.no.digits]]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 1 to 99, excluding start/stop characters and check digit. They should satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

[Ex] To enable the BHT to scan Code 128 with min.no.digits 6 and max.no.digits 12

BHT_EnableBar(TEXT("FL"), TEXT("K:6-12"))

[Ex] To enable the BHT to scan Code 128 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40

BHT_EnableBar(TEXT("FL"), TEXT("K:6-10,K:20-40"))

NOTE: Neither start/stop characters nor check digits will be transferred to the barcode buffer.

Handling special characters

If the BHT reads any barcode consisting of special characters only (such as FNC, CODEA, CODEB, CODEC and SHIFT characters), it will not transfer the data to the barcode buffer. The beeper sounds only if it is enabled.

Details about FNC characters

(1) FNC1

The BHT will not transfer an FNC1 character placed at the first or second character position immediately following the start character, to the barcode buffer. FNC1 characters in any other positions will be converted to GS characters (1Dh) and then transferred to the barcode buffer like normal data.

If an FNC1 immediately follows the start character, the barcode will be recognized as EAN-128 and marked with W instead of K.

(2) FNC2

If the BHT reads a barcode containing an FNC2 character(s), it will not buffer such data but transfer it excluding the FNC2 character(s).

(3) FNC3

If the BHT reads a barcode containing an FNC3 character(s), it will regard the data as invalid and transfer no data to the barcode buffer, while it may drive the indicator LED and beeper (vibrator) if activated this **BHT_EnableBar** function.

(4) FNC4

An FNC4 converts data encoded by the code set A or B into a set of extended ASCII-encoded data (128 added to each official ASCII code value).

1 A single FNC4 character converts only the subsequent data character into the extended ASCII-encoded data.

A pair of FNC4 characters placed in successive positions converts all of the subsequent data characters preceding the next pair of FNC4 characters or the stop character, into the extended ASCII-encoded data. If a single FNC4 character is inserted in those data characters, however, it does not convert the subsequent data character only.

An FNC4 character does not convert any of GS characters converted by an FNC1 character, into the extended ASCII-encoded data.

- Standard 2 of 5 (STF) (H)

Syntax

H[:[mini.no.digits[-max.no.digits]][CD][startstop]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BhtShell.exe) up to 99 digits.

If only max.no.digits is omitted, only the number of digits specified by mini.no.digits can be read.

CD is a check digit. Specifying a C to CD makes the Interpreter check barcodes with MOD-10. The check digit is included in the number of digits.

startstop specifies the normal or short format of the start/stop characters.

Specify N for the normal format; specify S for the short format. If startstop is omitted, start/stop characters can be read in either format.

[Ex] To enable the BHT to scan STF with min.no.digits 6 and max.no.digits 12

BHT_EnableBar(TEXT("FL"), TEXT("H:6-12"))

[Ex] To enable the BHT to scan STF with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40

BHT_EnableBar(TEXT("FL"), TEXT("H:6-10,H:20-40"))

□ MSI (P)

Syntax

P[:[mini.no.digits[-max.no.digits]][CD]]

Where

mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

$$\text{mini.no.digits} \leq \text{max.no.digits}$$

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C1 or C2 to CD makes the Interpreter check barcodes with a single-digit or two-digit CD, respectively. If no CD is specified, the Interpreter checks barcodes with a single-digit CD. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan MSI with min.no.digits 6, max.no.digits 12, and a single CD check
BHT_EnableBar(TEXT("FL"), TEXT("P:6-12C1"))

[Ex] To enable the BHT to scan MSI with min.no.digits 6, max.no.digits 10 and a single CD check or with min.no.digits 20, max.no.digits 40 and a two-digit CD check
BHT_EnableBar(TEXT("FL"),TEXT("P:6-10,P:20-40C2"))

□ RSS (R)

Syntax

R

RSS-14, RSS-14 Stacked, RSS Limited, RSS Expanded, and RSS Expanded Stacked codes can be read.

BHT-700Q

The readable barcodes include, among 2D codes, QR code, PDF417, MaxiCode, Data Matrix, and EAN·UCC composite, and, among barcodes, universal product code, interleaved 2of5 (ITF), Codabar (NW-7), Code 39, Code 93, Code 128, and RSS. Further, the BHT-700Q can read EAN-128 with Code 128 (read specified). (For details of readable codes, refer to the Instruction Manual.)

□ QR Codes (Q)

Syntax

Q [: [symbol type[min. code version [-max. code version]]][split code scanning mode]
[:symbol type[min code version[-max code version]]]
[:symbol type[min code version[-max code version]]]]

For symbol type, the following values are available:

Symbol type	Readable code
S	Micro QR
M	QR model 1
L	QR model 2

If you omit the symbol type, you can read Micro QR, QR model 1, and QR model 2.

The minimum and maximum code versions refer to those of QR code that can be read. The table below shows the possible ranges by symbol type.

Allowable range of code version	Symbol type
1 – 4	S
1 – 22	M
1 – 40	L

The minimum and maximum code versions must satisfy the following relationship:

Minimum code version ≤ Maximum code version

If you omit both the minimum and maximum code versions, you can read QR codes of a full range (up to the maximum allowable) of code versions for each symbol type. If you omit only the maximum code version, you can read only the QR code of the minimum code version you specify.

In split code scanning mode, you can read QR code symbols that are split into a maximum of 16 segments (sub-codes). You can specify any of the edit mode, batch edit mode, and non-edit mode as shown below.

Split code scanning mode	
E	Enable in edit mode
B	Enable in batch edit mode
C	Enable in non-edit mode

If you specify "E," "B," and "C," the latest specification takes effect.

If you do not specify the split code scanning mode, the BHT cannot read split QR code symbols.

[Ex] To enable the BHT to read split codes:

BHT_EnableBar (TEXT ("FB"), TEXT ("Q:M5-14E;L1-40;S1-4"))

In scanning a split code in edit mode, the maximum data length is 8,192 bytes. Data exceeding 8,192 bytes causes a read error to be recognized and the beeper to sound for 500 ms. The read data will be destroyed.

When a split code is read in non-edit mode, the read data is stored into the barcode buffer in the following format:

Sub-code no	No. of sub-codes	Parity	Read data
-------------	------------------	--------	-----------

Sub-code no., No. of sub-codes: 1 byte (hex.) (0 – F)

Parity: 2 bytes (hex.) (00 – FF)

The sub-code number, number of sub-codes, and parity are converted into hexadecimal characters.

The sub-code number is expressed in hexadecimal notation; for example, 0 (30h) for the first, and F (46h) for the 16th. Likewise, the number of sub-codes is expressed in hexadecimal notation; for example, 1 (31h) for the splitting into 2 segments, and F (46h) for the splitting into 16 segments.

The parity is provided for sum checking of the read data. It also serves as the delimiter between a group of split codes from another group.

In split code scanning, the beeper sounds as follows: Upon reading the first split code of a QR code, it beeps twice, signaling the start of the split code scanning mode. Thereafter, the beeper sounds once each time a split code is read, except the last one, which causes the beeper to sound three times, signaling the end of the split code scanning mode.

All split codes belonging to a QR code must be read, no matter what sequence it may be. Once read, a split code cannot be read again until all the other split codes belonging to the other QR code have been read.

In any of the following events, the split code scanning will be terminated, even if the scanning of all split codes of the QR code is not complete. If scanning is terminated in this manner in edit mode, all the data that has been read up to that point will be destroyed.

- A non-split code has been read:

In this case, the data that has been read will be stored into the barcode buffer.

- A split code belonging to another QR code has been read:

The BHT initiates the reading of the new sequence of split code scanning.

- The barcode reading window has been put away from the barcode for more than 3 seconds in the momentary switch mode, alternate switch mode, or continuous read mode; or more than 5 seconds has elapsed since a split code was read.
- The illuminating LED has been turned OFF by a trigger switch, i.e., in the momentary switch mode or auto-off mode, the trigger switch has been released, or in the alternate switch mode, the trigger switch has been pressed again.

□ PDF417(Y)

Syntax

Y [:[symbol type]]

For symbol type, you can specify one of the values shown below.

Symbol type	Applicable code
S	MicroPDF417
M	PDF417

If you do not specify the symbol type, both MicroPDF417 and PDF417 can be read.

□ MaxiCode(X)

Syntax

X

□ MaxiCode(Z)

Syntax

Z[:[symbol type [min code no.[-max code no.]]]
[;symbol type [min code no.[-max code no.]]]

For symbol type, you can specify one of the values shown below.

Symbol type	Applicable code
S	Square Data Matrix
R	Rectangular Data Matrix

“min code no.” and “max code no.” are the minimum and maximum DataMatrix code numbers that can be read, respectively. The table below shows the allowable range of code numbers by symbol type.

Allowable range of code number	Symbol type
1 to 24	S
1 to 6	R

If you do not specify the symbol type, both Square Data Matrix and Rectangular Data Matrix can be read.

“min code no.” and “max code no.” must satisfy the following relationship:

$$\text{min code no.} \leq \text{max code no.}$$

If you omit both the minimum and maximum code numbers, you can read DataMatrix codes of a full range (up to the maximum allowable) of code numbers for each symbol type. If you omit only the maximum code number, you can read only the DataMatrix code of the minimum code number you specify. The table below shows the correspondence between the code number and the number of cells.

S (Square Data Matrix)

Code No	ROWxCOL	Code No	ROWxCOL	Code No	ROWxCOL	Code No	ROWxCOL
1	10x10	7	22x22	13	44x44	19	88x88
2	12x12	8	24x24	14	48x48	20	96x96
3	14x14	9	26x26	15	52x52	21	104x104
4	16x16	10	32x32	16	64x64	22	120x120
5	18x18	11	36x36	17	72x72	23	132x132
6	20x20	12	40x40	18	80x80	24	144x144

R (Rectangular Data Matrix)

Code No	ROWxCOL	Code No	ROWxCOL
1	8x18	4	12x36
2	8x32	5	16x36
3	12x26	6	16x48

□ EAN·UCC Composite(V)

Syntax

V

- Universal product code (A)

Syntax

A [:[code]][1st character [2nd character]] [supplemental]]

Specify one of the codes listed below.

Code	Barcode type
A	EAN-13 (JAN-13), UPC-A
B	EAN-8 (JAN-8)
C	UPC-E

If you do not specify any of the codes, all of the above-listed codes can be read.

The first and second characters are the first characters representing the country flag and must be a numeral (0 through 9) each. A question mark (?) serves as a wild card.

“supplemental” refers to the reading of an add-on code. Specifying “S” as “supplemental” enables the BHT to read add-on codes.

To specify multi-line code reading, first specify “&” and then specify this syntax as many times as the number of rows to be read. The code cannot be omitted.

For multi-line code reading, refer to the section on multi-line code reading.

[Ex] Reading 3 rows of a universal product code:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,A:A,A:A:B,A:C"))

□ Interleaved 2of5 (ITF) (I)

Syntax

I [:[min no. digits [-max no. digits]][CD][:[1st character [2nd character]]]

“min no. digits” and “max no. digits” are the minimum and maximum numbers of digits of the barcode. You can specify any pair of numbers between 2 and 99 (inclusive) that satisfy the following relationship:

$$\text{min no. digits} \leq \text{max no. digits}$$

If you omit both the minimum and maximum numbers of digits, the BHT can read barcodes whose lengths are between the minimum number of digits specified in system mode and 99 (inclusive). If you omit only the maximum number of digits, the BHT can read only barcodes of the length specified by “min no. digits.”

“CD” represents the check digit. If you specify “C,” the barcode will be checked according to MOD-10. The check digit(s) is (are) included in the number of digits.

To specify multi-line code reading, first specify “&” and then specify this syntax as many times as the number of rows to be read. In this syntax, “;” and the portion after it are valid only in the case of multi-line code reading. Specify a numeral (0 – 9) in the first and second characters. For multi-line code reading, refer to the section on multi-line code reading.

[Ex] Reading two rows of an ITF code:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,I;;12,I;;23"))

□ Codabar (NW-7) (N)

Syntax

N [:[min no. digits [- max no. digits]][startstop] [CD]]

“min no. digits” and “max no. digits” are the minimum and maximum numbers of digits of the barcode. You can specify any pair of numbers between 3 and 99 (inclusive) that satisfy the following relationship:

$$\text{min no. digits} \leq \text{max no. digits}$$

If you omit both the minimum and maximum numbers of digits, the BHT can read barcodes whose lengths are between the minimum number of digits specified in system mode and 99 (inclusive). If you omit only the maximum number of digits, the BHT can read only barcodes of the length specified by “min no. digits.”

“startstop” means the start character and the stop character. Specify A, B, C, or D. A question mark (?) serves as a wild card. The start and stop characters are included in the number of digits. “A” through “D” are stored in the barcode buffer as “a” through “d.”

“CD” represents the check digit. If you specify “C,” the barcode will be checked according to MOD-16. The check digit(s) is (are) included in the number of digits.

To specify multi-line code reading, first specify “&” and then specify this syntax as many times as the number of rows to be read. For multi-line code reading, refer to the section on multi-line code reading.

[Ex] Reading 3 rows of a Codabar:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,N:8,N:6,N:4"))

□ Code 39 (M)

Syntax

M [:[min no. digits [-max no. digits]][CD];[1st character [2nd character]]]

“min no. digits” and “max no. digits” are the minimum and maximum numbers of digits of the barcode. The start character and the stop character are not included in the number of digits here. You can specify any pair of numbers between 1 and 99 (inclusive) that satisfy the following relationship:

$$\text{min no. digits} \leq \text{max no. digits}$$

If you omit both the minimum and maximum numbers of digits, the BHT can read barcodes whose lengths are between 1 and 99 (inclusive). If you omit only the maximum number of digits, the BHT can read only barcodes of the length specified by “min no. digits.”

“CD” represents the check digit. If you specify “C,” the barcode will be checked according to MOD-43. The check digit(s) is (are) included in the number of digits.

To specify multi-line code reading, first specify “&” and then specify this syntax as many times as the number of rows to be read. In this syntax, “;” and the portion after it are valid only in the case of multi-line code reading. Specify a numeral (0 – 9) in the first and second characters. For multi-line code reading, refer to the section on multi-line code reading.

[Ex] Reading 2 rows of a Code 39:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,M::12,M::23"))

□ CODE-93 (L)

Syntax

L [: [min.no.digits [-max.no.digits]][;1st digit [-2nd digit]]]

Where

min.no.digits and max.no.digits are the minimum and maximum numbers of digits for barcodes to be read by the BHT, respectively. These should be numerals from 1 to 99, excluding start/stop characters and check digits, and should satisfy the following condition:

$$\text{min no. digits} \leq \text{max no. digits}$$

If both the min.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only the max.no.digits is omitted, the BHT can only read the number of digits specified by min.no.digits.

[Ex.]To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 12:

BHT_EnableBar (TEXT ("FL"), TEXT ("L:6-12"))

[Ex.]To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 10, or min.no.digits 20 and max.no.digits 40:

BHT_EnableBar (TEXT ("FL"), TEXT ("L:6-10, L:20-40"))

NOTE: Neither start/stop characters nor check digits will be transferred to the barcode buffer.

To specify multi-line code reading, specify "&" followed by the above syntax as many times as the number of rows to be read. In the above syntax, information following ";" is valid only for multi-line code reading. Specify 0 to 9 for the 1st and 2nd digits. (Refer to the section on multi-line code reading for further details of specifying multi-line code reading.)

[Ex.] To read 2 rows of Code 93:

BHT_EnableBar (TEXT ("FB"), TEXT ("&, L:;12, L:;23"))

□ Code 128 (K)

Syntax

K [:[min no. digits [-max no. digits]][;[1st character [2nd character]]]]

“min no. digits” and “max no. digits” are the minimum and maximum numbers of digits of the barcode. The start character and the stop character are not included in the number of digits here. You can specify any pair of numbers between 1 and 99 (inclusive) that satisfy the following relationship:

$$\text{min no. digits} \leq \text{max no. digits}$$

If you omit both the minimum and maximum numbers of digits, the BHT can read barcodes whose lengths are between 1 and 99 (inclusive). If you omit only the maximum number of digits, the BHT can read only barcodes of the length specified by “min no. digits.”

The start character, the stop character, and the check digit are not stored into the barcode buffer.

To specify multi-line code reading, first specify “&” and then specify this syntax as many times as the number of rows to be read. In this syntax, “;” and the portion after it are valid only in the case of multi-line code reading. Specify a numeral (0 – 9) in the first and second characters. For multi-line code reading, refer to the section on multi-line code reading.

[Ex] Reading 2 rows of a Code 128:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,K::12,K::23"))

Positions of special characters

When a code consisting only of special characters (FNC, CODEA, CODEB, CODEC, and SHIFT characters) or a code containing FNC3 has been read, the read data is not stored into the barcode buffer. When beeper sounding is enabled, the beeper sounds.

Handling of FNC characters

(1) FNC1

The FNC1 character located 1 or 2 places after the start character will not be stored into the barcode buffer. An FNC1 character located elsewhere will be converted into a GS character (1Dh) and stored into the barcode buffer.

A code in which an FNC character immediately follows the start character is EAN-128, in which case the code mark is "W" instead of "K."

(2) FNC2

For a barcode containing an FNC2 character, the data will not be temporarily stored. Instead, the data code excluding the FNC2 character will be stored into the barcode buffer.

(3) FNC3

If a barcode contains an FNC3 character, the read data will be regarded as invalid and will not be stored into the barcode buffer. When the indicator LED and the vibrator are enabled by the **BHT_EnableBar** function, the indicator LED and the vibrator will be turned ON.

(4) FNC4

The FNC4 character converts data encoded by code set A or B into the extended ASCII format (normal ASCII + 128). One FNC4 character converts one data character immediately following it into the extended ASCII format.

A pair of contiguous FNC4 characters converts into the extended ASCII format all the data characters following it before another pair of contiguous FNC4 characters or a stop character. An exception is when a stand-alone FNC4 character exists in this string of characters, in which case one data character immediately following it will not be converted.

Also, the GS character created from an FNC1 character will not be converted into the extended ASCII format.

■ Multi-line code reading

To specify Multi-line code reading, specify "&" followed by the codes to be read. Up to three rows can be specified.

Syntax

"&, (code in 1st row), (code in 2nd row)[, (code in 3rd row)]"

The codes supported in multi-line code reading are the universal product code, interleaved 2of5 (ITF), Codabar (NW-7), Code 39, and Code 128 (all among barcodes).

(1) Multi-line code reading is independent of single-row code reading.

[Ex] Reading universal product code EAN-8 and EAN-13 (2 rows):

BHT_EnableBar (TEXT ("FB"), TEXT ("&,A:B,A:A"))

[Ex] Reading 1 row of universal product code EAN-8 and 2 rows of Code 39:

BHT_EnableBar (TEXT ("FB"), TEXT ("A:B,&,M,M"))

(2) You can specify a 2D code and a multi-line code simultaneously.

[Ex] Reading a QR code and 3 rows of code 39:

BHT_EnableBar (TEXT ("FB"), TEXT ("Q,&,M,M,M"))

(3) In Multi-line code reading, you can specify the reading sequence using the first two characters (start/stop in the case of Codabar).

[Ex] Reading 3 rows of ITF (with character specification) in the following sequence: code beginning with "12," code with CD beginning with "21" of 6 – 10 digits in length, and code beginning with "23" of 12 digits in length

BHT_EnableBar (TEXT ("FB"), TEXT ("&,I:;12,I:6-10C;21,I:12;23"))

You can also specify a single character.

[Ex] Reading a universal product code EAN and ITF (with character specification) in the following sequence: EAN beginning with "49" and ITF beginning with "2" of 6 – 10 digits in length.

BHT_EnableBar (TEXT ("FB"), TEXT ("&,A:A49,I:6-10;2"))

(4) Data will be output in the specified sequence.

[Ex] Data is to be output in the sequence of EAN-8 beginning with "12" - EAN-8 beginning with "21."

BHT_EnableBar (TEXT ("FB"), TEXT ("&,A:B12,A:B21"))

Note, however, that if you specify the same character and the same number of digits, then the output sequence is unpredictable.

[Ex] Reading 2 rows of ITF, both beginning with "49" and having a length of 6 digits:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,I:6;49,I:6;49"))

In this example, it is unpredictable, for example, which will be output first, ITF "495678" or ITF "498765."

- (5) If the same code (with the same code type and the same data code) appears more than once in a multi-line code, the BHT cannot read it.

[Ex] A code consisting of EAN-13: "4912345678904" in the first row, EAN-13; "1200000000003" in the second row, and EAN-13 "4912345678904" in the third row cannot be read with the following instruction:

BHT_EnableBar (TEXT ("FB"), TEXT ("&,A:A49,A:A12,A:A49"))

- (6) If you specify the same code type, the same number of digits, and the same conditions for single-row reading and multi-line code reading, the BHT cannot read the single-row code.

[Ex] If you have a single-row EAN-13 code "4901234567894" and a two-row EAN-13 code consisting of "4909876543214" in the first row and "1200000000003" in the second row, you cannot read them using the following instruction:

BHT_EnableBar (TEXT ("FB"), TEXT ("A:A49,&,A:A49,A:A12"))

- (7) In multi-line code reading, an ITF code less than 4 digits in length cannot be read unless you specify the number of digits.

- (8) You cannot specify multiple-row code reading for add-on codes in the universal product code.

- (9) You cannot specify multiple-row code reading for the RSS code.

- (10) When you have selected the point scan mode, you cannot specify multiple-row code reading. [

□ RSS (R)

Syntax

R

BHT_DisableBar

Description:

Close the barcode device file to disable barcode reading.

Syntax:

DWORD BHT_DisableBar (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened

BHT_ReadBar

Description

Read out data read from the barcode buffer.

If the string length longer than that of the read barcode is specified to dwBarLen, the remaining area following the read barcode will be filled with NULL codes.

If barcode reading is not enabled, an error (ERROR_INVALID_HANDLE) will result.

Syntax:

```
DWORD BHT_ReadBar (  
TCHAR* pwchBuffer ,  
DWORD dwBarLen ,  
DWORD* pdwActualBarLen )
```

Parameters

pwchBuffer

[out] Heading address of the storage buffer storing the read data

dwBarLen

[in] Maximum length of data to be read

pdwActualBarLen

[out] Length of data read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened.
ERROR_INVALID_PARAMETER	No storage address specified.

BHT_ReadBarEx

Description

Read out data from the barcode buffer and encodes it into the specified data format.

If the length of the read data is shorter than the specified maximum data length (dwBarLen), the excess part will be filled with 0s.

If barcode reading is disabled, an error (ERROR_INVALID_HANDLE) will be caused.

Syntax:

```
DWORD BHT_ReadBarEx (  
    DWORD dwDataType,  
    LPVOID lpBuffer,  
    DWORD dwBarLen,  
    DWORD* pdwActualBarLen )
```

Parameters

dwDataType

[in] Encoding format

 READ_CODE_BINARY : binary data (no encoding)

 READ_CODE_UNICODE : unicode data

lpBuffer

[in] Starting address of the read data in the storage buffer

dwBarLen

[in] Maximum read data length (maximum length of data to be read out)

pdwActualBarLen

[out] Length of data read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened.
ERROR_INVALID_PARAMETER	The specified encoding is wrong. No storage address specified.

BHT_GetBarType

Description

BHT-700B

Read the barcode type and the number of digits of a barcode read most recently.
If no barcode has been read after the BHT was turned on, the function gets "0."

BHT-700Q

Read the barcode type and the number of digits of a barcode read most recently.
If no barcode has been read after the BHT was turned on, the function gets "0."

When a multiple-row code has been read, this fact is communicated to the caller and the total number of digits in the multiple-row code is returned.

To get the information for a specific row, call **BHT_GetBarInfo**.

When an EAN·UCC composite code has been read, this fact is communicated to the caller and the total number of digits in the EAN·UCC composite code is returned. To get the information for a specific row, call **BHT_GetBarInfo**.

Syntax

```
DWORD BHT_GetBarType (  
  DWORD* pdwBarMark ,  
  DWORD* pdwBarlen )
```

Parameters

pdwBarMark

[out] Address for storing the barcode type

pdwBarlen

[out] Address for storing the barcode length

The *pdwBarMark* contains one of the following letters representing code types:

Barcode type	<i>pdwBarMark</i>
(No code read)	0
EAN-13 (JAN-13), UPC-A	'A'
EAN-8 (JAN-8)	'B'
UPC-E	'C'
ITF	'I'
STF (Only for BHT-700B)	'H'
CODABAR (NW-7)	'N'
CODE-39	'M'
CODE-93	'L'
CODE-128	'K'
EAN-128	'W'
MSI (Only for BHT-700B)	'P'
RSS	'R'
QR code (Only for BHT-700Q)	'Q'
Split QR code (in non-edit mode) (Only for BHT-700Q)	'S'
PDF417 (Only for BHT-700Q)	'Y'
Maxi Code (Only for BHT-700Q)	'X'
Data Matrix (Only for BHT-700Q)	'Z'
Multi-line code (Only for BHT-700Q)	'&'
Composite (Only for BHT-700Q)	'V'

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

BHT_GetBarInfo

Description

BHT-700B

Read the information on the code read most recently, including the code type and the number of digits in the code.

If no barcode has been read after the BHT was turned on, the function gets "0" for both the code type and the number of digits.

BHT-700Q

Read the information on the code read most recently, including the code type and the number of digits in the code.

If no barcode has been read after the BHT was turned on, the function gets "0" for both the code type and the number of digits.

When a multi-line code has been read, the information on all the rows is obtained in an array format. Also, the number of rows in the code is obtained.

When an RSS-EAN Composite code has been read, the information on all the codes constituting the composite code is obtained in an array format. Also, the number of codes in the composite code is obtained.

Syntax

```
DWORD BHT_GetBarInfo (  
    ST_CODE_INFO* pstInfo ,  
    DWORD* pdwCodeNum )
```

Parameters

pstInfo

[out] Destination address into which the code information is to be stored

pdwCodeNum

[in] No. of codes to be obtained

[out] Destination address into which the number of codes is to be stored. This is set to "1" when a code other than a multiple-row code or an EAN-UCC composite code has been read.

Shown below is the format of the structure containing code information. For the relationship between dwType and code type, refer to **BHT_GetBarType**.

```
struct ST_CODE_INFO {  
    DWORD dwType; // code type  
    DWORD dwLen; // no. of digits  
};
```

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

If you specify NULL in pstCodeInfo, the number of elements of ST_CODE_INFO necessary to store the read code will be stored into pdwCodeNum.

An error occurs if a value greater than **MAX_NUM_CODE_1D_SCANNER** (when using the BHT-700B) or **MAX_NUM_CODE_2D_SCANNER** (when using the BHT-700Q) is specified for pdwCodeNum.

BHT_GetBarNum

Description

Read the number of digits of the barcode remaining in the barcode buffer.
If barcode reading is not enabled, an error (ERROR_INVALID_HANDLE) will result.

Syntax

```
DWORD BHT_GetBarNum (  
DWORD* pdwCodeNum )
```

Parameters

pdwCodeNum

[out] Address for storing the barcode length

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened
ERROR_INVALID_PARAMETER	Storage address not specified

BHT_GetBarChkDgt

Description

Calculate a check digit (CD) of the barcode data according to the calculation method specified by dwCDType.

Syntax

```
DWORD BHT_GetBarChkDgt (  
TCHAR* pwchBarbuf ,  
DWORD dwCDType ,  
DWORD* pdwChkdgt )
```

Parameters

pwchBarbuf

[in] Heading address of barcode data storage buffer

dwCDType

[in] Check digit type

Barcode type and the corresponding calculation method

Barcode Type	dwCDType	Calculation Method
EAN(JAN), UPC	'A'	MOD10 (Modulo arithmetic-10)
ITF	'I'	MOD10 (Modulo arithmetic-10)
STF (only for BHT-700B)	'H'	MOD10 (Modulo arithmetic-10)
CODABAR (NW-7)	'N'	MOD16 (Modulo arithmetic-16)
CODE-39	'M'	MOD43 (Modulo arithmetic-43)
MSI (only for BHT-700B)	'P'	MOD10 (Modulo arithmetic-10)

pdwChkdgt

(out) Address for storing the check digit calculated

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid check digit type. Invalid barcode data. Storage address not specified.

Comment:

If barcode data contains a character(s) out of the specification of the barcode type specified by dwCDType, then this function sets "0" and returns an error code. However, if only the CD position character in barcode data is out of the specification, this function calculates the correct CD and returns it as one-character string.

[Ex 1] **BHT_GetBarChkDgt**(TEXT("494AB4458"), 'A', &dwChkDgt);

"A" and "B" are out of the specification of EAN or UPC, so dwChkDgt is "0" and the function returns an error code.

[Ex 2] **BHT_GetBarChkDgt**(TEXT("4940045X"), 'A', &dwChkDgt);

"X" is out of the specification but it is a CD position character, so this function calculates the correct CD and dwChkDgt is "8."

[Ex 3] **BHT_GetBarChkDgt**(TEXT("a0ef3-a"), 'N', &dwChkDgt);

"e" and "f" are out of the specification of Codabar (NW-7), so dwChkDgt is "0" and the function returns an error code.

[Ex 4] **BHT_GetBarChkDgt**(TEXT("a123Qa"), 'N', &dwChkDgt)

"Q" is out of the specification but it is a CD position character, so this function calculates the correct CD and dwChkDgt is "-."

When dwCDType is A (EAN or UPC), this function identifies the EAN or UPC depending upon the data length (number of digits) as listed below. If the data length is a value other than 13, 8, and 7, this function gets "0" and returns an error code.

Data length of barcode data	Barcode type
13	EAN-13 (JAN-13), UPC-A
8	EAN-8 (JAN-8)
7	UPC-E

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("49400458"), 'A', &dwChkDgt);
if ( dwChkDgt == '8' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("4940045"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("0"));
BHT_GetBarChkDgt(wchBarData1, 'A', &dwChkDgt);
wprintf(TEXT("CD = %s%c"), wchBarData, dwChkDgt);
```

Result

> CD = 49400458

When dwCDType is I (ITE), the length of barcode data must be an even number of two or more digits. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("123457"), 'I', &dwChkDgt);  
if ( dwChkDgt == '7' ) {  
    printf("CD OK");  
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("12345"));  
wcscpy(wchBarData1, wchBarData);  
wcscat(wchBarData1, TEXT("0"));  
BHT_GetBarChkDgt(wchBarData1, 'I', &dwChkDgt);  
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

> CD = 123457

When dwCDType is H (STF), the length of barcode data must be two or more digits. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("12345678905"), 'H', &dwChkDgt);  
if ( dwChkDgt == '5' ) {  
    printf("CD OK");  
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("1234567890"));  
wcscpy(wchBarData1, wchBarData);  
wcscat(wchBarData1, TEXT("5"));  
BHT_GetBarChkDgt(wchBarData1, 'H', &dwChkDgt);  
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

> CD = 12345678905

When dwCDType is N (Codabar), the length of barcode data must be three digits or more including start and stop characters. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("a0123-a"), 'M', &dwChkDgt);
if ( dwChkDgt == '-' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcsncpy(wchBarData, TEXT("a0123a"));
len = wcslen(wchBarData);
wcsncpy(wchTmp1BarData, wchBarData, len - 1);
wcsncpy(wchTmp2BarData, wchTmp1BarData);
wscat(wchTmp2BarData, TEXT("0"));
wscat(wchTmp2BarData, &(wchBarData[len - 1]));
BHT_GetBarChkDgt(wchTmp2BarData) 'M', &dwChkDgt);
wprintf(TEXT("%s%c%s"), wchTmp1BarData, dwChkDgt, &wchTmp2BarData[len-1]);
```

Result

> CD = a0123-a

When dwCDType is M (Code 39), the length of barcode data must be two or more digits except for start and stop characters. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("CODE39W"), 'M', &dwChkDgt);  
if ( dwChkDgt == 'W' ) {  
    printf("CD OK");  
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("CODE39"));  
wcscpy(wchBarData1, wchBarData);  
wcscat(wchBarData1, TEXT("0"));  
BHT_GetBarChkDgt(wchBarData1, 'M', &dwChkDgt);  
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

> CD = CODE39W

When dwCDType is P (MSI), the length of barcode data must be two or more digits. If not, this function gets "0" and returns an error code. To calculate a two-digit CD, call this function twice.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("123456782"), 'P', &dwChkDgt);  
if (dwChkDgt == '2' ) {  
    printf("CD OK");  
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("12345678"));  
wcscpy(wchBarData1, wchBarData);  
wcscat(wchBarData1, TEXT("0"));  
BHT_GetBarChkDgt(wchBarData1, 'P', &dwChkDgt);  
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

> CD = 123456782

BHT_BAR_SetDecodeOptions

Description

Sets the editing function setting value for the decoded result.
This is supported only by the BHT-700B.

Syntax

```
DWORD BHT_BAR_SetDecodeOptions (  
    EN_DCD_OPTIONS_CODE_TYPE enCodeType,  
    LPVOID pOptions,  
    DWORD dwLen)
```

Parameters

enCodeType

[in] Code type to be edited

pOptions

[out] Editing function setting value. The addresses for the structure are listed below.

Code type	<i>EnCodeType</i>	<i>pOptions</i>
UPC-E	EnOptionsUPCE	ST_DCD_UPCE_OPTIONS
UPC-A	EnOptionsUPCA	ST_DCD_UPCA_OPTIONS
EAN-8	enOptionsEAN8	ST_DCD_EAN8_OPTIONS

dwLen

[in] *pOptions* size (bytes). Sets the value calculated at `Sizeof`.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	The storage address has not been set. Invalid size specified.

Construction

The member construction for **EN_DCD_OPTIONS_CODE_TYPE** used to specify the code type is as follows.

```
typedef enum _EN_DCD_OPTIONS_CODE_TYPE {  
    enOptionsUPCE,  
    enOptionsUPCA,  
    enOptionsEAN8,  
} EN_DCD_OPTIONS_CODE_TYPE;
```

The member construction for **ST_DCD_UPCE_OPTIONS**, **ST_DCD_UPCA_OPTIONS**, and **ST_DCD_EAN8_OPTIONS** is shown below.

```
typedef struct _ST_DCD_UPCE_OPTIONS {  
    BOOL    bConvertToUPCA;  
    BOOL    bReportNumsys;  
    BOOL    bReportChk;  
} ST_DCD_UPCE_OPTIONS, *PST_DCD_UPCE_OPTIONS;
```

Member name	Default	Details
<i>bConvertToUPCA</i>	FALSE	Used to convert (TRUE) or not convert (FALSE) to UPC-A.
<i>bReportNumsys</i>	FALSE	Used to add (TRUE) or not add (FALSE) a "0" at the beginning.
<i>bReportChk</i>	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_UPCA_OPTIONS {
    BOOL    bReportNumsys;
    BOOL    bReportChk;
} ST_DCD_UPCA_OPTIONS, *PST_DCD_UPCA_OPTIONS;
```

Member name	Default	Details
<i>bReportNumsys</i>	TRUE	Used to add (TRUE) or not add (FALSE) a "0" at the beginning.
<i>bReportChk</i>	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_EAN8_OPTIONS {
    BOOL    bConvertToEAN13;
} ST_DCD_EAN8_OPTIONS, *PST_DCD_EAN8_OPTIONS;
```

Member name	Default	Details
<i>bConvertToEAN13</i>	FALSE	Used to convert (TRUE) or not convert (FALSE) to EAN-13.

Notes

Authorize reading of the code type prior to conversion when authorizing code reading with **BHT_EnableBar**.

The value acquired with **BHT_ReadBar**, **BHT_GetBarType**, **BHT_GetBarNum**, and **BHT_GetBarInfo** will be the value after conversion.

The set value will only be valid within the application in which it is set. Settings are not updated to other applications.

(Ex.) The following settings are used in order to convert UPC-E codes to UPC-A codes.

```
ST_DCD_UPCE_OPTIONS stOptions;
DWORD dwLen = sizeof(stOptions);
BHT_EnableBar(TEXT("FB"), TEXT("A:C"));
.....
...
/* Acquires current setting */
BHT_BAR_GetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, &dwLen);
/* Authorizes conversion to UPC-A */
stOptions.bConvertToUPCA = TRUE;
BHT_BAR_SetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, dwLen);
```

BHT_BAR_GetDecodeOptions

Description

Sets the editing function setting value for the decoded result.
This is supported only by the BHT-700B.

Syntax

```
DWORD BHT_BAR_GetDecodeOptions (  
  EN_DCD_OPTIONS_CODE_TYPE enCodeType,  
  TCHAR * pwchSysParam,  
  LPVOID pOptions,  
  DWORD* pdwLen)
```

Parameters

enCodeType

[in] Code type to be edited

pOptions

[out] Editing function setting value. The addresses for the structure are listed below.

Code type	<i>enCodeType</i>	<i>pOptions</i>
UPC-E	enOptionsUPCE	ST_DCD_UPCE_OPTIONS
UPC-A	enOptionsUPCA	ST_DCD_UPCA_OPTIONS
EAN-8	enOptionsEAN8	ST_DCD_EAN8_OPTIONS

pdwLen

[in] pOptions size (bytes). Sets the value calculated at sizeof.

[out] Size of valid data stored in pOptions (bytes).

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	The storage address has not been set. Invalid size specified.

Construction

The member construction for **EN_DCD_OPTIONS_CODE_TYPE** used to specify the code type is as follows.

```
typedef enum _EN_DCD_OPTIONS_CODE_TYPE {  
  enOptionsUPCE,  
  enOptionsUPCA,  
  enOptionsEAN8,  
} EN_DCD_OPTIONS_CODE_TYPE;
```

The member construction for **ST_DCD_UPCE_OPTIONS**, **ST_DCD_UPCA_OPTIONS**, and **ST_DCD_EAN8_OPTIONS** is shown below.

```
typedef struct _ST_DCD_UPCE_OPTIONS {  
  BOOL   bConvertToUPCA;  
  BOOL   bReportNumsys;  
  BOOL   bReportChk;  
} ST_DCD_UPCE_OPTIONS, *PST_DCD_UPCE_OPTIONS;
```


Member name	Default	Details
<i>bConvertToUPCA</i>	FALSE	Used to convert (TRUE) or not convert (FALSE) to UPC-A.
<i>bReportNumsys</i>	FALSE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
<i>bReportChk</i>	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_UPCA_OPTIONS {
    BOOL    bReportNumsys;
    BOOL    bReportChk;
} ST_DCD_UPCA_OPTIONS,
*PST_DCD_UPCA_OPTIONS;
```

Member name	Default	Details
<i>bReportNumsys</i>	TRUE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
<i>bReportChk</i>	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_EAN8_OPTIONS {
    BOOL    bConvertToEAN13;
} ST_DCD_EAN8_OPTIONS,
*PST_DCD_EAN8_OPTIONS;
```

Member name	Default	Details
<i>bConvertToEAN13</i>	FALSE	Used to convert (TRUE) or not convert (FALSE) to EAN-13.

Notes

The acquired value will be the value set at that application.

(Ex.) The following settings are used in order to convert UPC-E codes to UPC-A codes.

```
ST_DCD_UPCE_OPTIONS stOptions;
DWORD dwLen = sizeof(stOptions);
BHT_EnableBar(TEXT("FB"), TEXT("A:C"));
.....
...
/* Acquires current setting */
BHT_BAR_GetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, &dwLen);
/* Authorizes conversion to UPC-A */
stOptions.bConvertToUPCA = TRUE;
BHT_BAR_SetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, dwLen);
```

16.2. Backlight API

BHT_SetBltStatus

Description

Control the backlight.

Syntax

```
DWORD BHT_SetBltStatus (  
    DWORD dwStatus )
```

Parameters

dwStatus

[in] Backlight status

<i>dwStatus</i>	Specification
BHT_BL_ENABLE_ON(*1)	Turn on the backlight.
BHT_BL_ENABLE_OFF	Turn off the backlight.
BHT_BL_DISABLE	Disable the backlight.

(*1) The backlight specified with the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_DEVICE,...) function illuminates.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

BHT_GetBltStatus

Description

Read the backlight status.

Syntax

```
DWORD BHT_GetBltStatus (  
    DWORD* pdwStatus )
```

Parameters

pdwStatus

[out] Current backlight status

<i>pdwStatus</i>	Specification
BHT_BL_ENABLE_ON	Backlight ON
BHT_BL_ENABLE_OFF	Backlight OFF
BHT_BL_DISABLE	Backlight enabled

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

16.3. Battery API

BHT_GetPowerStatus

Description

Read information about the battery loaded in the BHT body.

Syntax

```
DWORD BHT_GetPowerStatus (  
WORD* pwCuOnLine ,  
WORD* pwBatteryFlag ,  
WORD* pwBatteryVoltage ,  
WORD* pwBatteryChemistry )
```

Parameters

pwCuOnLine

[out] Read the BHT state on/off the CU

<i>pwCuOnLine</i>	Specification
AC_LINE_ONLINE	Placed on the CU
AC_LINE_OFFLINE	Not placed on the CU

pwBatteryFlag

[out] Read battery voltage level

<i>pwBatteryFlag</i>	Specification
BHT_BATTERY_FLAG_HIGH	High level ($3.9\text{ V} \leq \text{Voltage}$)
BHT_BATTERY_FLAG_MID	Medium level ($3.7\text{ V} \leq \text{Voltage} < 3.9\text{ V}$)
BHT_BATTERY_FLAG_LOW	Low level ($3.6\text{ V} \leq \text{Voltage} < 3.7\text{ V}$)
BHT_BATTERY_FLAG_WARNING	Warning level ($\text{Voltage} < 3.6\text{ V}$)
BHT_BATTERY_FLAG_CRITICAL	Critical level ($\text{Voltage} < 3.5\text{ V}$)
BHT_BATTERY_FLAG_NO_BATTERY	No battery loaded

pwBatteryVoltage

[out] Battery output voltage (mV)

pwBatteryChemistry

[out] Battery type

<i>pwBatteryChemistry</i>	Specification
BATTERY_CHEMISTRY_LION	Lithium ion battery
BATTERY_CHEMISTRY_UNKNOWN	Unknown

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

Comments

"BHT_BATTERY_FLAG_NO_BATTERY" for the battery level are never actually acquired.

BHT_GetPowerStatus2nd

Description

Read information about the battery loaded in the grip.

Syntax

```
DWORD BHT_GetPowerStatus2nd (  
WORD* pwCuOnLine ,  
WORD* pwBatteryFlag ,  
WORD* pwBatteryVoltage ,  
WORD* pwBatteryChemistry )
```

Parameters

pwCuOnLine

[out] Read the BHT state on/off the CU

<i>pwCuOnLine</i>	Specification
AC_LINE_ONLINE	Placed on the CU
AC_LINE_OFFLINE	Not placed on the CU

pwBatteryFlag

[out] Read battery voltage level

<i>pwBatteryFlag</i>	Specification
BHT_BATTERY_FLAG_NO_BATTERY	No battery loaded or no grip connected

pwBatteryVoltage

[out] Battery output voltage (mV)
"0" is always returned.

pwBatteryChemistry

[out] Battery type

<i>pwBatteryChemistry</i>	Specification
BATTERY_CHEMISTRY_UNKNOWN	Unknown

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

Comments

With the BHT-700, the battery is not stored in the grip, and therefore the following information is returned if this function is called.

- Battery voltage level : BHT_BATTERY_FLAG_NO_BATTERY (No battery loaded)
- Battery output voltage : 0 mV
- Battery type : BATTERY_CHEMISTRY_UNKNOWN (Unknown)

16.4. LED API

BHT_GetNLedStatus

Description

Read the status of the indicator LED (red/blue).

Syntax

```
DWORD BHT_GetNLedStatus (  
    DWORD* pdwInfo )
```

Parameters

pdwInfo

[out] Address for storing the LED status

<i>pdwInfo</i>	Specification
LED_OFF	Both red and blue LEDs OFF
RED_LED_ON	Red LED ON
GREEN_LED_ON	Blue LED ON
RED_LED_ON GREEN_LED_ON	Both red and blue LEDs ON

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

BHT_SetNLedStatus

Description

Control the indicator LED (red/blue).

Syntax

```
DWORD BHT_SetNLedStatus (  
    DWORD dwStatus )
```

Parameters

dwStatus

[in] Controls the LED ON/OFF

<i>dwStatus</i>	Specification
LED_OFF	Turn off both red and blue LEDs
RED_LED_ON	Turn on red LED only
GREEN_LED_ON	Turn on blue LED only
RED_LED_ON GREEN_LED_ON	Turn on both red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_EnableBar**, the LED can be controlled.
- If the LED is turned on by this function in a user program, it will be kept on until this function turns off the LED even if the user program is terminated.

BHT_GetNLedStatusEx

Description

Read the status of the indicator LED and synchronization LED.

Syntax

```
DWORD BHT_GetNLedStatusEx (  
    DWORD dwLedDevice ,  
    DWORD* pdwStatus )
```

Parameters

dwLedDevice
[in] LED device

<i>dwLedDevice</i>	Specification
LED_BAR	Indicator LED

pdwStatus
[out] Address for storing the LED status

<i>pdwStatus</i>	Specification
	If <i>dwLedDevice</i> = LED_BAR
RED_LED_ON	Red LED ON (Blue LED OFF)
GREEN_LED_ON	Blue LED ON (Red LED OFF)
RED_LED_ON GREEN_LED_ON	Both red and blue LEDs ON

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified.

BHT_SetNLedOn

Description

Turn on the indicator LED and/or synchronization LED.

Syntax

```
DWORD BHT_SetNLedOn (  
    DWORD dwLedDevice ,  
    DWORD dwLedNum )
```

Parameters

dwLedDevice

[in] LED device

<i>dwLedDevice</i>	Specification
LED_BAR	Indicator LED

dwLedNum

[in] LEDs to be turned on

<i>dwLedNum</i>	Specification
	If <i>dwLedDevice</i> = LED_BAR
RED_LED	Red LED
GREEN_LED	Blue LED
RED_LED GREEN_LED	Red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_EnableBar**, the LED can be controlled.
- If the LED is turned on by this function in a user program, it will be kept on until this function turns off the LED even if the user program is terminated.

BHT_SetNLedOff

Description

Turn off the indicator LED and/or synchronization LED.

Syntax

```
DWORD BHT_SetNLedOff (  
    DWORD dwLedDevice ,  
    DWORD dwLedNum )
```

Parameters

dwLedDevice
[in] LED device

<i>dwLedDevice</i>	Specification
LED_BAR	Indicator LED

dwLedNum
[in] LEDs to be turned off

<i>dwLedNum</i>	Specification
	If <i>dwLedDevice</i> = LED_BAR
RED_LED	Red LED
GREEN_LED	Blue LED
RED_LED GREEN_LED	Red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_DisableBar**, the LED can be controlled.

16.5. Beeper/Vibrator API

BHT_StartBeep

Description

Drive the beeper or vibrator.

Syntax

```
DWORD BHT_StartBeep (  
    DWORD dwOnTime ,  
    DWORD dwOffTime ,  
    WORD wRepCnt ,  
    WORD wFreq )
```

Parameters

dwOnTime

[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime

[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt

[in] Number of repetitions, Entry range: 0 to 255

wFreq

[in] Frequency (Hz) , Entry range: 0 to 32767

Specification of 0, 1 or 2 to *wFreq* produces the special beeper effects as listed below.

<i>wFreq</i>	Tone	Frequency (Hz)
0	Low-pitched	698
1	Medium-pitched	1396
2	High-pitched	2793

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:

- The system functions allow the beeper volume to be changed. (Refer to Section 5.2. and 5.3.)
- Specification of any of 3 through 667 to wFreq deactivates the beeper or vibrator.
- Specification of zero to dwOnTime deactivates the beeper or vibrator.
- Specification of a value except zero to dwOnTime and wRepCnt and specification of zero to dwOffTime keep the beeper sounding.
- For your reference, the relationship between the frequencies and the musical scale is listed below.

	Scale 1	Scale 2	Scale 3	Scale 4
do (C)	-	1046	2093	4186
do# (C#)	-	1108	2217	
re (D)	-	1174	2349	
re# (D#)	-	1244	2489	
mi (E)	-	1318	2637	
fa (F)	698	1396	2793	
fa# (F#)	739	1479	2959	
sol (G)	783	1567	3135	
sol# (G#)	830	1760	3520	
la (A)	880	1760	3520	
la (A#)	932	1864	3729	
si (B)	987	1975	3951	

BHT_StartBeeperOnly

Description

Drive the beeper.

Syntax

```
DWORD BHT_StartBeeperOnly (  
    DWORD dwOnTime ,  
    DWORD dwOffTime,  
    WORD wRepCnt ,  
    WORD wFreq )
```

Parameters

dwOnTime

[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime

[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt

[in] Number of repetitions, Entry range: 0 to 255

wFreq

[in] Frequency (Hz) , Entry range: 0 to 32767

Specification of 0, 1 or 2 to *wFreq* produces the special beeper effects as listed below.

<i>wFreq</i>	Tone	Frequency (Hz)
0	Low-pitched	698
1	Medium-pitched	1396
2	High-pitched	2793

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:

- The system functions allow the beeper volume to be changed. (Refer to Section 5.2. and 5.3.)
- Specification of any of 3 through 667 to wFreq deactivates the beeper or vibrator.
- Specification of zero to dwOnTime deactivates the beeper or vibrator.
- Specification of a value except zero to dwOnTime and wRepCnt and specification of zero to dwOffTime keep the beeper sounding.
- For your reference, the relationship between the frequencies and the musical scale is listed below.

	Scale 1	Scale 2	Scale 3	Scale 4
do (C)	-	1046	2093	4186
do# (C#)	-	1108	2217	
re (D)	-	1174	2349	
re# (D#)	-	1244	2489	
mi (E)	-	1318	2637	
fa (F)	698	1396	2793	
fa# (F#)	739	1479	2959	
sol (G)	783	1567	3135	
sol# (G#)	830	1760	3520	
la (A)	880	1760	3520	
la (A#)	932	1864	3729	
si (B)	987	1975	3951	

BHT_StartVibrationOnly

Description

Drive the vibrator.

Syntax

```
DWORD BHT_StartVibrationOnly (  
    DWORD dwOnTime ,  
    DWORD dwOffTime ,  
    WORD wRepCnt )
```

Parameters

dwOnTime

[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime

[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt

[in] Number of repetitions, Entry range: 0 to 255

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

16.6. Wireless Communication API

BHT_RF_Open

Description

Open the wireless LAN device and enable wireless communication.

Syntax

DWORD BHT_RF_Open (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_SHARING_VIOLATION	Bluetooth device is opened.

Remarks

Wireless LAN and Bluetooth device cannot be opened at the same time. If wireless LAN device tries to be opened while Bluetooth device is opened, an error (ERROR_SHARING_VIOLATION) is returned.

BHT_RF_OpenEx

Description

Sets the communication format, opens the wireless LAN device and enables wireless communication.

Syntax

```
DWORD BHT_RF_OpenEx (  
    DWORD dwOpt )
```

Parameters

dwOpt

[in] Communication format

<i>dwOpt</i>	Specification
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_INVALID_PARAMETER	Parameter error
ERROR_SHARING_VIOLATION	Bluetooth device is opened.

Remarks

Wireless LAN and Bluetooth device cannot be opened at the same time. If wireless LAN device tries to be opened while Bluetooth device is opened, an error (ERROR_SHARING_VIOLATION) is returned.

BHT_RF_Close

Description

Close the wireless LAN device and disable wireless communication.

Syntax

```
DWORD BHT_RF_Close ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion

BHT_RF_CloseEx

Description

Closes the wireless LAN device for the set format and disables wireless communication.

Syntax

```
DWORD BHT_RF_CloseEx (  
DWORD dwOpt )
```

Parameters

dwOpt

[in] Communication format

<i>dwOpt</i>	Specification
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error

BHT_RF_IoControl

Description

Sends a control command to the driver and performs an operation corresponding to that command.

Syntax

```
DWORD BHT_RF_IoControl (  
    DWORD Oid ,  
    LPVOID lpInBuf ,  
    DWORD nInBufSize ,  
    LPVOID lpOutBuf ,  
    DWORD nOutBufSize ,  
    LPVOID lpBytesReturned )
```

Parameters

Oid

[in] Control command ID

<i>Oid</i>	Specification
RF_UPDATE_PROFILE	Updates the profile settings for the BHT wireless registry. (*1)
RF_COMMIT_PROFILE	Updates the changed parameter value to the driver. (*2)
RF_SET_PROFILE	Selects the profile to be edited.
RF_REMOVE_PROFILE	Deletes the profile.
RF_GET_PROFILE_COUNT	Acquires the number of completed profiles.
RF_GET_PROFILE_KEY	Acquires the profile key.

(*1) Copies values set at the ZeroConfig GUI to the BHT wireless registry referenced by the wireless driver.

(*2) Updates values set at this API to ZeroConfig.

lpInBuf

[in] Header address for buffer in which input data is stored

nInBufSize

[in] Size of buffer in which input data is stored (Bytes)

lpOutBuf

[out] Header address for buffer in which output data is stored

nOutBufSize

[out] Size of buffer in which output data is stored (Bytes)

lpBytesReturned

[out] Size of actually acquired output data (Bytes)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Storage header address unset
ERROR_NOT_READY	Can not access ZeroConfig service
ERROR_NOT_ENOUGH_MEMORY	The number of profiles has exceeded the maximum (16).
ERROR_NOT_FOUND	The relevant profile cannot be found.
ERROR_FILE_NOT_FOUND	The relevant file cannot be found.

The details set for each argument differ for each command.

<i>Old</i>	<i>lpInBuf</i>	<i>nInBufSize</i>	<i>lpOutBuf</i>	<i>nOutBufSize</i>
RF_UPDATE_PROFILE	–	–	–	–
RF_COMMIT_PROFILE	–	–	–	–
RF_SET_PROFILE	ST_RF_PROFILE_KEY (*3)	ST_RF_PROFILE_KEY size	–	–
RF_REMOVE_PROFILE	ST_RF_PROFILE_KEY	ST_RF_PROFILE_KEY size	–	–
RF_GET_PROFILE_COUNT	–	–	Profile count storage variable	sizeof(DWORD)
RF_GET_PROFILE_KEY	Profile index to be acquired	sizeof(DWORD)	ST_RF_PROFILE_KEY	ST_RF_PROFILE_KEY size

(*3) Use ESSID and Infrastructure mode to specify the profile. Create a new profile if no profile can be found corresponding to the specified ESSID and Infrastructure mode.

The ST_RF_PROFILE_KEY configuration is as follows.

Construction

```
typedef struct _ST_RF_PROFILE_KEY {
    TCHAR szESSID [SSID_MAX+1]; // ESSID
    UCHAR ucReserved [2];        // reserved
    DWORD dwInfraMode;            // Infrastructure mode
} ST_RF_PROFILE_KEY;
```

Members

szESSID

SSID specified character string

dwInfraMode

Infrastructure mode

<i>dwInfraMode</i>	Specification
INFRA_INFRASTRUCTURE	Infrastructure

BHT_RF_Synchronize

Description

Get the association status.

Syntax

```
DWORD BHT_RF_Synchronize (  
    long ITimeout ,  
    long* pISync )
```

Parameters

ITimeout

[in] Timeout (in units of 100 ms)

<i>ITimeout</i>	Specification
> 0	Confirm the synchronization status until timeout
0	Check the synchronization status immediately and return the result
-1	Try to synchronize with the access point until synchronized

pISync

[out] Address for storing the synchronization result

<i>pISync</i>	Specification
0	Successfully synchronized
-1	Synchronization incomplete (timed out)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

BHT_RF_GetParamInt

Description

Read integer from the wireless communications parameter.

Syntax

```
DWORD BHT_RF_GetParamInt (  
    DWORD dwParam ,  
    DWORD* pdwData ,  
    DWORD* pdwLen )
```

Parameters

dwParam

[in] Parameter number

<i>dwParam</i>	Specification
P_INT_POWERSAVE	Power mode dwData = P_PWRSAVE_CAM = P_PWRSAVE_PSP
P_INT_RADIOMODE	Radio mode dwData = P_RADIOMODE_11A = P_RADIOMODE_11B = P_RADIOMODE_11B P_RADIOMODE_11G
P_INT_AUTHENTICATE	Authentication method dwData = P_AUTH_OPEN = P_AUTH_SHARED = P_AUTH_WPA = P_AUTH_WPA2 = P_AUTH_WPA2PSK
P_INT_ENCRYPTION	Encryption dwData = P_ENCRYPT_DISABLE = P_ENCRYPT_WEP = P_ENCRYPT_TKIP = P_ENCRYPT_AES
P_INT_8021X	802.1x authentication (EAP type) dwData = P_8021X_DISABLE = P_8021X_PEAP = P_8021X_TLS
P_INT_PRIORITY	Profile priority dwData = 1 (high) to 16 (low)
P_INT_INDEXKEY	Index key dwData = 1 to 4

pdwData

[out] Address for storing data obtained

pdwLen

[out] Address for storing the length of data obtained

If the function succeeds in getting data, the length of data obtained is always 4.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Address for storing data obtained not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_SetParamInt

Description

Write integer to the wireless communications parameter.

Syntax

```
DWORD BHT_RF_SetParamInt (  
    DWORD dwParam ,  
    const DWORD* pdwData ,  
    DWORD dwLen )
```

Parameters

dwParam

[in] Parameter number

<i>dwParam</i>	Specification
P_INT_POWERSAVE	Power mode dwData = P_PWRSAVE_CAM = P_PWRSAVE_PSP
P_INT_RADIOMODE	Radio mode dwData = P_RADIOMODE_11A = P_RADIOMODE_11B = P_RADIOMODE_11B P_RADIOMODE_11G
P_INT_AUTHENTICATE	Authentication method dwData = P_AUTH_OPEN = P_AUTH_SHARED = P_AUTH_WPA = P_AUTH_WPA2 = P_AUTH_WPA2PSK
P_INT_ENCRYPTION	Encryption dwData = P_ENCRYPT_DISABLE = P_ENCRYPT_WEP = P_ENCRYPT_TKIP = P_ENCRYPT_AES
P_INT_8021X	802.1x authentication (EAP type) dwData = P_8021X_DISABLE = P_8021X_PEAP = P_8021X_TLS
P_INT_PRIORITY	Profile priority dwData = 1 (high) to 16 (low)
P_INT_INDEXKEY	Index key dwData = 1 to 4

pdwData

[in] Destination address where the set data is to be stored

dwLen

[in] Length of data

The data length is always 4.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Address for storing data obtained not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_GetParamStr

Description

Read string from the wireless communications parameter.

Syntax

```
DWORD BHT_RF_GetParamStr (  
    DWORD dwParam ,  
    TCHAR* pwchData ,  
    DWORD* pdwLen )
```

Parameters

dwParam

[in] Parameter number

<i>dwParam</i>	Specification
P_STR_VERSION	Driver version
P_STR_FW_VERSION	Firmware version
P_STR_MACADDRESS	MAC address

pwchData

[out] Heading address of the storage buffer for data obtained

pdwLen

[out] Length of data obtained

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_SetParamStr

Description

Write character string to the wireless communications parameter.

Syntax

```
DWORD BHT_RF_SetParamStr (  
    DWORD dwParam ,  
    TCHAR* pwchData ,  
    DWORD dwLen )
```

Parameters

dwParam

[in] Parameter number

<i>dwParam</i>	Specification
P_STR_WEPKEY1	WEP Key 1
P_STR_PRESHAREDKEY	Pre Shared Key

pwchData

[in] Heading address of the storage buffer for data specified

dwLen

[in] Length of data specified

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error
ERROR_NOT_SUPPORTED	Not supported.

BHT_RF_GetInfoInt

Description

Read integer from the communications parameter.

Syntax

```
DWORD BHT_RF_GetInfoInt (  
    DWORD dwType ,  
    DWORD* pdwInfo )
```

Parameters

dwType

[in] Type of information to be read out

<i>dwType</i>	Specification
P_RATE_INFO	Current communication speeds: No link → P_RATE_NOT_LINK 1Mbps → P_RATE_1MBPS 2Mbps → P_RATE_2MBPS 5.5Mbps → P_RATE_5_5MBPS 11Mbps → P_RATE_11MBPS Above 11Mbps → P_RATE_OVER11MBPS
P_RATE_INFO2	Current communication speeds (Units: 100bps): [Ex.] 5.5Mbps → 55,000 11Mbps → 110,000 54Mbps → 540,000
P_CHANNEL_INFO	Frequency channel currently used

pdwInfo

[out] Address for storing info read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

BHT_RF_GetInfoStr

Description

Read string from the communications parameter.

Syntax

```
DWORD BHT_RF_GetInfoStr (  
    DWORD dwType ,  
    TCHAR* pwchInfo )
```

Parameters

dwType

[in] Type of information to be read out

<i>dwType</i>	Specification
P_APMAC_INFO	MAC address of AP being linked

pwchInfo

[out] Heading address of the storage buffer for info read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

BHT_RF_GetSiteSurvey

Description

Get the quality of the communications link.

Syntax

```
DWORD BHT_RF_GetSiteSurvey (  
    DWORD* pdwStrength ,  
    DWORD* pdwBeacon ,  
    DWORD* pdwLink )
```

Parameters

pdwStrength

[out] Current signal strength, 0 to 100 (%)

pdwBeacon

[out] The same value as *pdwStrength*, 0 to 100 (%)

pdwLink

[out] Current link quality

<i>pdwLink</i>	Specification
LQ_UNSYNC	Not associated
LQ_POOR	Poor communications link (less than 26%)
LQ_FAIR	Fair communications link (26% or more and less than 42%)
LQ_GOOD	Good communications link (42% or more and less than 74%)
LQ_EXCELLENT	Excellent communications link (74% or more for send and receive)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
No NIC device found.	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

16.7. OS Updating API

BHT_SystemModify

Description

Update the BHT OS.

Syntax

```
DWORD BHT_SystemModify (  
    DWORD dwCtrlCode,  
    TCHAR * pwchSysParam,  
    DWORD dwLen,  
    DWORD * pdwLenReturned )
```

Parameters

pwszFileName

[in] Pointer filename that points a NULL-appended character string containing the OS reconfiguration filename.

dwMode

[in] Reboot mode after turning the power off

<i>dwMode</i>	Specification
SYSMDFY_POWEROFF	Turn the power off. (Cold-boot the BHT at the next power on)
SYSMDFY_REBOOT	Perform a cold boot.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_FILE_NOT_FOUND	Specified file or device not found. (OS reconfiguration file not found.)
ERROR_INVALID_PARAMETER	Parameter error.
ERROR_BAD_FORMAT	The OS update file is incorrect.

16.8. Bluetooth API

BHT_BT_PowerOn

Description

Turns ON the Bluetooth device power supply and enables Bluetooth.

Syntax

DWORD BHT_BT_PowerOn (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.
ERROR_SHARING_VIOLATION	Wireless LAN device is opened.

Remarks

Wireless LAN and Bluetooth device cannot be opened at the same time. If Bluetooth device tries to be opened while wireless LAN device is opened, an error (ERROR_SHARING_VIOLATION) is returned.

BHT_BT_PowerOff

Description

Turns OFF the Bluetooth device power supply and disables Bluetooth.

Syntax

DWORD BHT_BT_PowerOff (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.

BHT_BT_GetPowerStatus

Description

Acquires the Bluetooth device power status.

Syntax

```
DWORD BHT_BT_GetPowerStatus (  
    DWORD *pdwStatus )
```

Parameters

pdwStatus

[in] Device status storage location address

The following values are returned for the device status.

<i>pdwStatus</i>	Specification
BHT_BT_POWER_ON	The Bluetooth device power is ON.
BHT_BT_POWER_OFF	The Bluetooth device power is OFF.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.
ERROR_INVALID_PARAMETER	Storage location address unset

16.9. Other APIs

BHT_WaitEvent

Description

Make the system wait until the specified event or timeout occurs.

Syntax

```
DWORD BHT_WaitEvent (  
    DWORD dwEvtNum ,  
    DWORD dwEvtMask ,  
    DWORD dwTimeOut ,  
    DWORD* pdwSignalEvent )
```

Parameters

dwEvtNum

[in] Number of events to wait

dwEvtMask

[in] Waiting event mask

<i>dwEvtMask</i>	Specification
EVT_MASK_KEYDOWN	Key depressed
EVT_MASK_TRGDOWN	Trigger switch depressed
EVT_MASK_TCHUP	Stylus released
EVT_MASK_DECODE	Decoding completed
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception (IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)

NOTE: ORing these events enables the BHT to wait for the two or more events.

dwTimeOut

[in] Timeout period (ms)

pdwSignalEvent

[out] Address for storing an event mask that occurred

<i>pdwSignalEvent</i>	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception(IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)
EVT_MASK_TIMEOUT	Timeout

NOTE: To make the system wait for occurrence of any event infinitely, specify INFINITE in *dwTimeOut*.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified

Comment:

The following five types of events can be specified:

- Depression of any key
- Depression of the trigger switch
- Stylus release
- Decoding completion
- Data reception (in IrDA interface, Serial interface, USB interface)

Specifying two or more events concurrently using this function allows the system to wait for occurrence of any of these events. To wait for other events in addition to events listed above, add desired events using macros with the event names defined by the BHTLIB.h library.

[Ex] Wait for occurrence of entry by any key depression or decoding completion for 10 seconds

```
BHT_WaitEvent (2, EVT_MASK_KEYDOWN | EVT_MASK_DECODE,  
10 * 1000, &dwSignalEvent);
```

BHT_WaitStandbyEvent

Description

Make the system wait until the specified event occurs.

Syntax

```
BHT_WaitStandbyEvent (  
    DWORD dwEvtNum ,  
    DWORD dwEvtMask ,  
    DWORD* pdwSignalEvent )
```

Parameters

dwEvtNum

[in] Number of events to wait

dwEvtMask

[in] Events to wait

<i>dwEvtMask</i>	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE	Data reception(IrDA interface)
EVT_MASK_RECEIVE_IRDA	
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)

pdwSignalEvent

[out] Address for storing events that occurred

<i>pdwSignalEvent</i>	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE	Data reception(IrDA interface)
EVT_MASK_RECEIVE_IRDA	
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified

Comment:

The following five types of events can be specified:

- Depression of any key
- Depression of the trigger switch
- Stylus release
- Decoding completion
- Data reception (in IrDA interface, Serial interface, USB interface)

Unlike **BHT_WaitEvent**, this function lets the CPU enter the standby mode when making the system wait, reducing power consumption. Note that execution of any other active thread will be suspended during execution of this function.

BHT_ShutdownSystem

Description

Turn off the BHT and boot the BHT according to the mode specified by the parameter.

Syntax

```
DWORD BHT_ShutdownSystem (  
DWORD dwMode )
```

Parameters

dwMode

[in] Power-off mode

<i>dwMode</i>	Specifications
BHT_PWR_WARM	Turn off and warm-boot the BHT. No power-off action is required. The contents in the RAM can be retained.
BHT_PWR_SUSPEND	Transfer control to the suspended mode. Pressing the power key starts the BHT. The contents in the RAM will be retained as long as the sub-battery is charged.
BHT_PWR_COLD_REGINIT	Turn off and cold-boot the BHT. Pressing the power key starts the BHT. The contents in the RAM will be lost and the system registry will be initialized.
BHT_PWR_COLD_REGREMAIN	Turn off and cold-boot the BHT. Pressing the power key starts the BHT. The contents of the system registry will be saved into the non-volatile memory in powering-off sequence and restored at the cold boot.
BHT_PWR_SYSMODIFY	A cold boot is performed automatically after turning OFF the power. With the BHT-700, this is the same as BHT_PWR_COLD .
BHT_PWR_COLD	A cold boot is performed automatically after turning OFF the power. If the registry has been saved, the BHT is booted based on the values for that registry, however, if it has not been saved, the BHT is booted based on the values for the default registry value.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:

Any of the following five modes can be specified:

- Warm boot*
- Suspend
- Cold boot* with Registry initialization (The Registry backup will also be lost.)
- Cold boot* without Registry initialization

- Cold boot*

*Contents of the memory after warm-/cold-booting the BHT

	After warm booting	After cold booting
Files in the FLASH folder	Retained	Retained
Files in the RAM	Retained	Erased
Contents of the Registry	Retained	Erased (Note)
Data being edited	Erased	Erased

(Note) If the Registry has been backed up, the backup will apply.

BHT_RegStore

Description

Save the registry.

Syntax

```
DWORD BHT_RegStore ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_WRITE_FAULT	Failed to save registry.

Chapter 17. Programming Using OCX (OLE Customer Control)

The BHT-700 Software Development Kit (BHT-700 SKD) provides ActiveX Control that can be used for programming applications for barcode reading and file transfer. This chapter gives information for using the ActiveX control.

17.1. System Requirements

- (1) BHT-700 Software Development Kit
- (2) Control files .ocx for the desktop
 - Scanner700.ocx: For barcode reading (for BHT-700B)
 - Scanner700Q.ocx: For barcode reading (for BHT-700Q)
 - FileTransfer700.ocx: For file transmission
 - FileTransferPC.ocx: For file transmission(for PC)

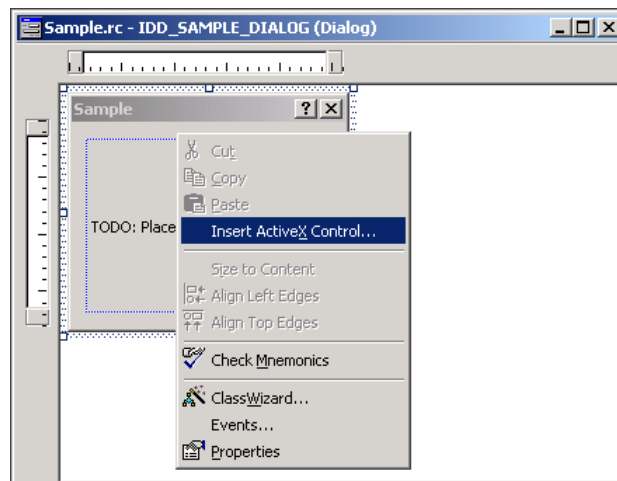
17.2. Installation

- (1) Copy the .ocx files in the BHT-700 Software Development Kit CD onto the appropriate folder of your PC.
- (2) Open the DOS command prompt and change the directory to the folder including the .ocx files.
- (3) Run the following two commands on the command line (>):
 - > regsvr32 Scanner700.ocx
 - > regsvr32 Scanner700Q.ocx
 - > regsvr32 FileTransfer700.ocx
 - > regsvr32 FileTrrnaferPC.ocx

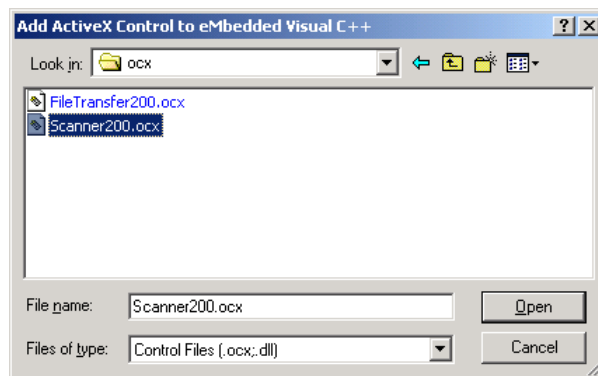
17.3. Using OCX

In Microsoft Foundation Class (MFC)

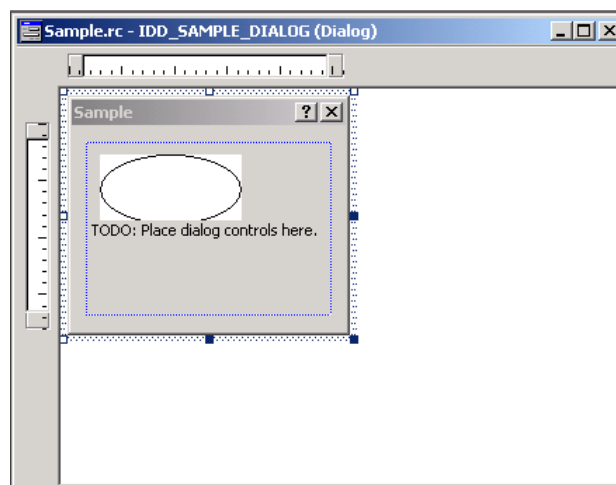
- (1) Open an existing project or create a new project in eMbedded Visual C++.
- (2) Insert the newly installed ActiveX control into eMbedded Visual C++. (This step is required only when the ActiveX control is first used after installation.)
- (3) -1 Point and right-click the active window or dialog, then choose "Insert ActiveX Control" command on the dropdown menu.



(2)-2 Click **Add Control** and choose the newly installed OCX by clicking **Open**.

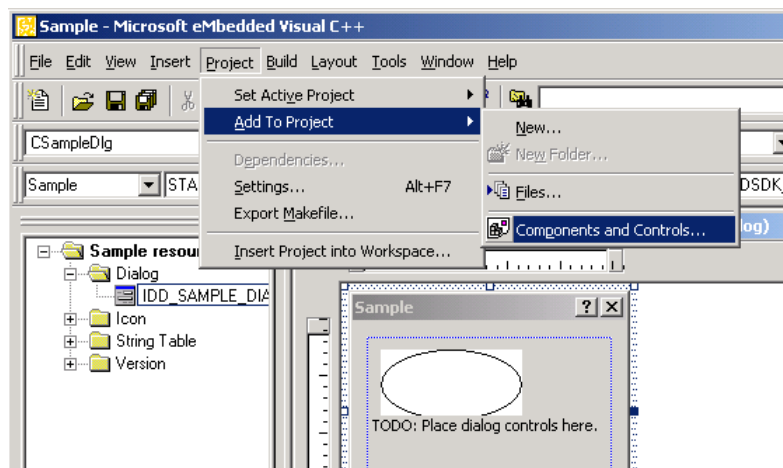


(2)-3 Click **OK**, and the control is pasted as shown below.

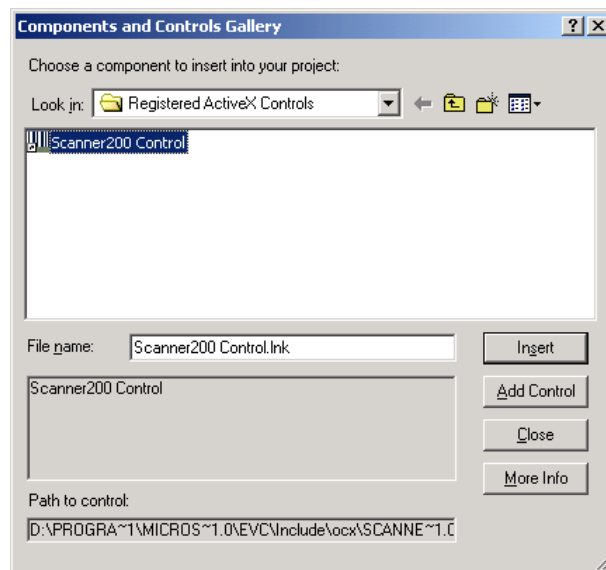


(3) Add the control to the project.

(3)-1 Click **Project–Add to Project–Components and Controls** on the menu bar as shown below.



(3)-2 Select the installed .OCX file.



(3)-3 Click **Insert**, and the message “Do you insert component?” pops up. Click **OK**, and specify an appropriate class name, header filename and implement filename.

(3)-4 If **OK** is clicked, an icon of the added control will be added to the dialog as shown below (red-circled).



(4) Following ClassWizard, assign a member variable to the inserted control.

17.4. Scanner Control

17.4.1. Properties

Name and type eVCpp		R/W	Value	Default value	Description
GetPortOpen SetPortOpen	BOOL	R/W	TRUE or FALSE	FALSE	Enable/disable flag for barcode reading TRUE: Enable FALSE: Disable
GetReadMode SetReadMode	CString	R/W	(*1)	"FB"	Character string for specifying the read mode (*1), (*2)
GetReadType SetReadType	CString	R/W	(*1)	<u>BHT700B</u> "A,I:4-99,M:1-99, N:3-99,L:1-99, K:1-99,H:3-99,P:1-99" <u>BHT700Q</u> " Q:E,A,I:4-99,L:1-99,M:1-99,N:3-99,K:1-99,Y,X,Z,R,V "	Character string for specifying the enable read code (*1), (*2)
GetBufferData SetBufferData	CString	R	-	""	Data stored in the barcode buffer (*1)
GetBufferCount SetBufferCount	long	R	-	0	Number of digits stored in the barcode buffer (*1)
GetBufferType SetBufferType	long	R	-	0	Barcode type stored in the barcode buffer (*1)
GetLastCount SetLastCount (*5)	long	R	-	0	Number of digits in the barcode read last
GetLastType SetLastType (*5)	long	R	-	0	Barcode type read last
GetLastCodeNum (*6)	long	R	-	0	No. of barcodes read last (*7)
GetErrorStatus SetErrorStatus	long	R/W	(*3)	ERROR_SUCCESS	Error code that occurred last (*4)
GetWaitStby SetWaitStby	BOOL	R/W	TRUE or FALSE	FALSE	Whether or not the control transfers to the standby mode before decoding completes TRUE: Transfer FALSE: Not transfer

(*1) Refer to **BHT_EnableBar** function.

(*2) Even if a value out of the range is specified, no error occurs. If TRUE is set to the portOpen property with the value being out of the range, an error occurs.

(*3) For details about error codes, refer to Section 17.4.4 Error Codes."

(*4) A new error code overwrites the old one whenever an error occurs. The ERROR_SUCCESS does not overwrite.

(*5) only for Scanner700.ocx

(*6) only for Scanner700Q.ocx

(*7) "1" when a code other than a multi-line code or a composite code has been read.

17.4.2. Methods

GetChkDigit

Description

Calculate a check digit (CD) of the barcode data according to the specified calculation method. (Refer to the **BHT_GetBarChkDgt** function.)

Syntax

```
long GetChkDigit (  
    TCHAR* BarData ,  
    short ChkDgtType )
```

Parameters

BarData

[in] Character string of the barcode

ChkDgtType

[in] Check digit type

(For details, refer to the **BHT_GetBarChkDgt** function.)

Return value

Value of the check digit calculated

GetLastCount

Description

Supported only on BHT-700Q

Read the number of digits in the specified row of the code that was read most recently.

Syntax

```
long GetLastCount (  
long CodeNo)
```

Parameters

CodeNo

[in] Row number for which you wish to get the number of digits (starting with “0” for the first row).

Return value

No. of digits in the row specified in *CodeNo*

If [the row number specified in *CodeNo* + 1] is larger than the number of rows actually read, “0” will be returned.

GetLastType

Description

Supported only on BHT-700Q

Read the code type in the specified row of the code that was read most recently.

Syntax

```
long GetLastType(  
long CodeNo)
```

Parameters

CodeNo

[in] Row number for which you wish to get the code type (starting with “0” for the first row).

Return value

Code type in the row specified in *CodeNo*

If [the row number specified in *CodeNo* + 1] is larger than the number of rows actually read, “0” will be returned.

17.4.3. Event Callback Function

DecodeDone

Description

This function is called when decoding is successfully completed. It reads out the `bufferData` property to get data decoded.

Syntax

```
void OnDecodeDone ( void )
```

Parameters

None

Return value

None

17.4.4. Error Codes

If an error occurs during access to properties or during calling to methods, the error code will be stored into the `errorStatus` variable.

Error Code Table

Propertie or Method	Name	Content
portOpen	ERROR_TOO_MANY_OPEN_FILES	Barcode reading enabled (when flag is TRUE).
	ERROR_INVALID_PARAMETER	readMode or readType out of the range (when flag is TRUE)
	ERROR_INVALID_HANDLE	Barcode reading disabled (when flag is FALSE)
BufferData	ERROR_INVALID_HANDLE	Barcode reading disabled
GetChkDigit	ERROR_INVALID_PARAMETER	Check digit type out of the range or invalid barcode data

17.4.5. Coding Sample

```
/* Initialize main dialog */
BOOL CBarOCXDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    .....
    .....
    /* Enable barcode reading */
    m_ScanCtrl.SetPortOpen(TRUE);

    return TRUE;
}

/* Initialize main dialog */
void CBarOCXDlg::OnDestroy()
{
    /* Disable barcode reading */
    m_ScanCtrl.SetPortOpen(FALSE);

    CDialog::OnDestroy();
}

/* Callback for decoding completion */
void CBarOCXDlg::OnDecodeDoneScannerctrl()
{
    CString BarData; /* Read data */

    /* Read data from buffer */
    BarData = m_ScanCtrl.GetBufferData();
    /* Display */
    .....
    .....
}
```

17.5. File Transfer Control

17.5.1. Properties

Name		R/W	Value	Default value	Content
eVC++					
GetPort SetPort	short	R/W	COM1 COM4	COM4	COM port
GetBaud SetBaud	long	R/W	CBR_300 (*1) CBR_600 (*1) CBR_1200 (*1) CBR_2400 (*1) CBR_4800 (*1) CBR_9600 CBR_19200 CBR_38400 CBR_57600 CBR_115200	CBR_115200	Transmission rate
GetParity SetParity	short	R/W	NOPARITY ODDPARITY (*1) EVENPARITY (*1)	NOPARITY	Parity
GetStopBit SetStopBit	short	R/W	ONESTOPBIT TWOSTOPBITS (*1)	ONESTOPBIT	Stop bit
GetPath SetPath	CString LPCTSTR	R/W	Absolute path starting with \ sign	"\"	Folder to store send files Folder to store receive files
GetTransferringEventInterval SetTransferringEventInterval	long	R/W	0 to 2147483647	0	Transferring Event interval during transmission (in units of 100 ms) 0 for no event
GetLinkTimeout SetLinkTimeout	long	R/W	0 to 65535	30 (30sec.)	Time required from commencement of transmission to timeout (in seconds) No timeout occurs when set to 0.
GetRetransmissionInterval SetRetransmissionInterval	long	R/W	1 to 65535	30 (30sec.)	Retransmission interval (in units of 100 ms)
GetTransmissionTimeout SetTransmissionTimeout	long	R/W	1 to 65535	30 (30sec.)	Time required for transmission timeout (in seconds)

(*1) Only for COM1

17.5.2. Methods

Function	Description
AddFile	Add a file to be transmitted.
ClearFile	Clear a file added by AddFile.
GetFile	Acquires the file name of the file to be transmitted or received.
GetFileCount	At the transmission side, returns the number of transmitted files, and at the receipt side, returns the number of received files, including the file currently being received.
GetTransferredCount	Returns the number of files for which transmission or receipt is complete.
Send	Transmit a file specified by AddFile.
Receive	Receive a file.
Abort	Abort the current file transmission process.
GetState	Get the current file transmission status.
GetError	Return the error information about the transaction processed last.

AddFile

Description

Add a file to be transmitted. Specify the filename excluding its pathname. The length of the filename is within 90 characters.

Syntax

```
long AddFile (  
LPCTSTR FileName )
```

Parameters

FileName

[in] Filename excluding pathname

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	NULL set to the parameter. Filename length is 0.
ERROR_FILENAME_EXCED_RANGE	Filename too long

ClearFile

Description

Clears a file added by AddFile.

Syntax

```
void ClearFile ( void )
```

Parameters

None

Return value

None

GetFile

Description

Acquires the file name of the file to be transmitted or received.

The maximum value given by the "Index" parameter is the number of files acquired with **GetFileCount**.

Syntax

```
CString GetFile (long Index)
```

Parameter

Index

[in] Index (1 or greater)

Return Value

File name of the specified Index (character string with length 0 when the Index lies outside the range).

GetFileCount

Description

At the transmission side, returns the number of transmitted files, and at the receipt side, returns the number of received files, including the file currently being received.

Syntax

```
short GetFileCount ( void )
```

Parameters

None

Return value

Transmission side:

Number of transmitted files (number of files added with **AddFile**)

Receipt side:

Number of received files (number of received files + file currently being transmitted)

Comment:

This value is cleared when **Receive** or **ClearFile** is called.

GetTransferredCount

Description

Returns the number of files for which transmission or receipt is complete.

Syntax

```
void GetTransferredCount (void)
```

Parameter

None

Return Values

Transmission side:

Number of files for which transmission is complete

Receipt side:

Number of files for which receipt is complete

Comment:

This value is cleared when **Receive** or **ClearFile** is called.

Send

Description

Transmit a file specified by AddFile.

Syntax

Long Send (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_ACCESS_DENIED	Access to COM port denied (e.g., occupied by other tasks)
ERROR_FILE_NOT_FOUND	Specified file or device not found
ERROR_NO_MORE_FILES	No send file found (No file added by AddFile .)
ERROR_BAD_PATHNAME	Path too long (Path + filename > 260 characters)

Receive

Description

Receive a file.

Syntax

long Receive (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_ACCESS_DENIED	Access to COM port denied (e.g., occupied by other tasks)
ERROR_FILE_NOT_FOUND	Specified file or device not found

Abort

Description

Abort the current file transmission process. After aborting, the *Done* event will occur.

Syntax

```
Void Abort ( void )
```

Parameters

None

Return value

None

GetState

Description

Get the current file transmission status.

Syntax

```
short GetState ( void )
```

Parameters

None

Return value

Error code	Meaning
TRANSFER_READY	On standby
TRANSFER_SEND	Transmitting
TRANSFER_RECEIVE	Receiving

GetError

Description

Return the error information for the transaction processed last.

Syntax

```
long GetError ( void )
```

Parameters

None

Return value

Code of an error that occurred during processing of methods.

17.5.3. Event Callback Functions

Function	Description
Done	This function is called when the transmission ends as specified.
Transferring	Get the information about a file being transmitted.

Done

Description

This function is called when the transmission ends as specified.

Syntax

```
void OnDone (  
long Result )
```

Parameters

Result

[out] End code listed in the table below

Result	Meaning
RROR_SUCCESS	Succeeded.
ERROR_TIMEOUT	Timeout.
ERROR_OPERATION_ABORTED	Process is aborted.
ERROR_OPEN_FAILED	Failed to open a file.
ERROR_INVALID_DATA	Invalid data received.
ERROR_DISK_FULL	Sufficient storage area not reserved.
ERROR_BAD_PATHNAME	Path too long (Path + filename > 260 characters)

Return value

None

Transferring

Description

Get the information about a file being transmitted.

Syntax

```
void OnTransferring (  
LPCTSTR FileName ,  
long Total ,  
long Transferred )
```

Parameters

FileName

[out] Name of file being transmitted

Total

[out] Size of file being transmitted

Transferred

[out] Size of file already transmitted

Return value

None

17.5.4. Coding Sample

```
void CSerialTransferDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CSerialTransferDlg)
    DDX_Control(pDX, IDC_FILETRANSFERCTRL1, m_clFileTransfer);
   //}}AFX_DATA_MAP
}

BEGIN_EVENTSINK_MAP(CSerialTransferDlg, CDialog)
   //{{AFX_EVENTSINK_MAP(CSerialTransferDlg)
    ON_EVENT(CSerialTransferDlg, IDC_FILETRANSFERCTRL1, 1 /* Done */, OnDoneFiletransferctrl, VTS_I4)
    ON_EVENT(CSerialTransferDlg, IDC_FILETRANSFERCTRL1, 2 /* Transferring */,
    OnTransferringFiletransferctrl, VTS_BSTR VTS_I4 VTS_I4)
   //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

/* Start download */
void CSerialTransferDlg::OnDownload()
{
    m_clFileTransfer.SetPath(TEXT("\\My Documents"));           // Set a filepath for the work file
    m_clFileTransfer.SetTransferringEventInterval(10);         // File transmission event (1s)
    m_clFileTransfer.Receive();                                  // Start transmission
}

/* Start upload */
void CSerialTransferDlg::OnUpload()
{
    m_clFileTransfer.SetPath(TEXT("\\My Documents"));           // Set a filepath for the work file
    m_clFileTransfer.AddFiles(TEXT("File1.dat"));               // Transmission file 1
    m_clFileTransfer.AddFiles(TEXT("File2.dat"));               // Transmission file 2
    m_clFileTransfer.AddFiles(TEXT("File3.dat"));               // Transmission file 3
    m_clFileTransfer.SetTransferringEventInterval(10);         // File transmission event (1s)
    m_clFileTransfer.Send();                                     // Start transmission
}

/* Abort */
void CSerialTransferDlg::OnAbort()
{
    m_clFileTransfer.Abort();                                    // Abort
}

/* Send/receive complete */
void CSerialTransferDlg::OnDoneFiletransferctrl(long Result)
{
    CString clMsg;
    clMsg.Format(TEXT("Done:%d"), Result);
    AfxMessageBox(clMsg, MB_ICONINFORMATION);
}

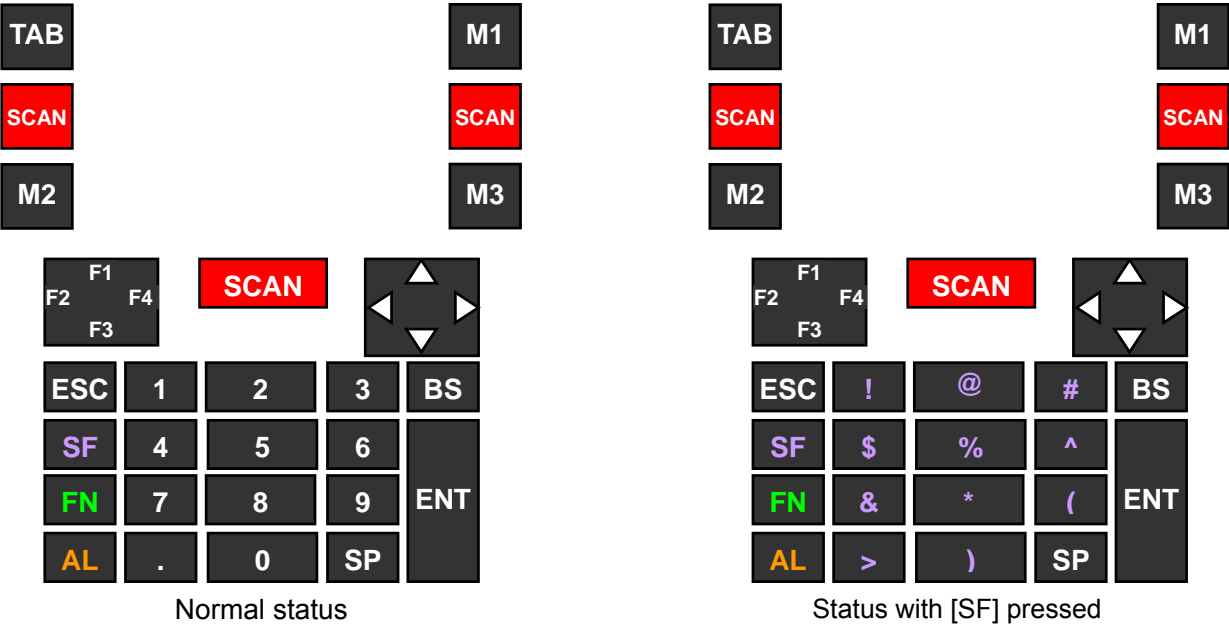
/* Display the info about file being transmitted */
void CSerialTransferDlg::OnTransferringFiletransferctrl(LPCTSTR FileName, long Total,
long Transferred)
{
    if(0 < Total)
    {
        TCHAR szProgress[MAX_PATH];
        wsprintf(szProgress, TEXT("%s %d%%"), FileName, (int)(Transferred*100/Total));
        SetWindowText(szProgress);                             // Display on the title bar
    }
}
```

Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes

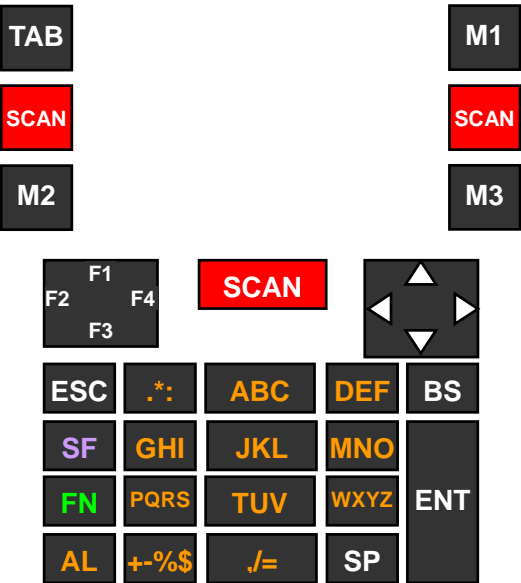
A.1. 27-key pad

A.1.1. Keyboard Arrangement

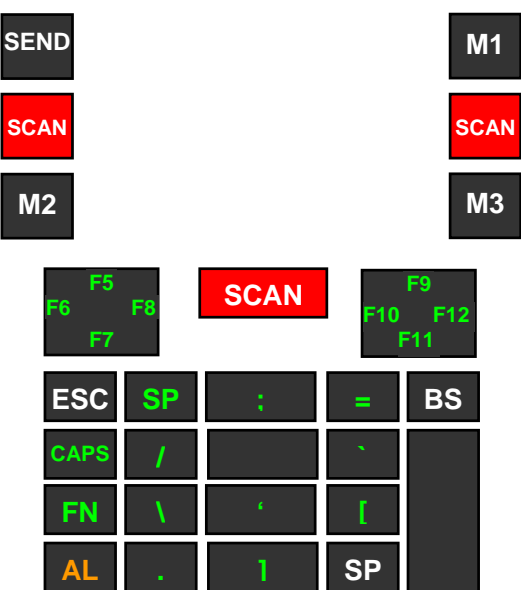
(1) Numeric entry mode



(2) Alphabet entry mode



(3) Function mode



A.1.2. Virtual Key Codes and Character Codes

Numeric entry mode

Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
	Constant	Value		Constant	Value	
[F1]	VK_F1	70	-	←	←	←
[F2]	VK_F2	71	-	←	←	←
[F3]	VK_F3	72	-	←	←	←
[F4]	VK_F4	73	-	←	←	←
[▲]	VK_UP	26	-	←	←	←
[▼]	VK_DOWN	28	-	←	←	←
[◀]	VK_LEFT	25	-	←	←	←
[▶]	VK_RIGHT	27	-	←	←	←
[9]	VK_9	39	39(9)	←	←	28()
[8]	VK_8	38	38(8)	←	←	2A(*)
[7]	VK_7	37	37(7)	←	←	26(&)
[6]	VK_6	36	36(6)	←	←	5E(^)
[5]	VK_5	35	35(5)	←	←	25(%)
[4]	VK_4	34	34(4)	←	←	24(\$)
[3]	VK_3	33	33(3)	←	←	23(#)
[2]	VK_2	32	32(2)	←	←	40(@)
[1]	VK_1	31	31(1)	←	←	21(!)
[0]	VK_0	30	30(0)	←	←	29())
[.]	VK_PERIOD	BE	2E(.)	←	←	3E(>)
[SP]	VK_SPACE	20	20()	←	←	←
[ESC]	VK_ESCAPE	1B	1B	←	←	←
[SF]	VK_SHIFT	10	-	←	←	←
[FN]	VK_FUNC	D2	-	←	←	←
[AL]	VK_ALP	D0	-	←	←	←
[BS]	VK_BACK	08	08	←	←	←
[ENT]	VK_RETURN	0D	-	←	←	←
[SCAN]	VK_SCAN	D1	-	←	←	←
[TAB]	VK_TAB	09	09	←	←	←
[M1]	VK_M1(*1)	C1(*1)	-(*1)	←	←	←
[M2]	VK_M2(*1)	C2(*1)	-(*1)	←	←	←
[M3]	VK_M3(*1)	C3(*1)	-(*1)	←	←	←

(*1) Virtual key codes and character codes will differ based on the key settings.

For details, refer to section "6.4 Magic Key Control"

Function mode

Key	Virtual key code		Character code
	Constant	Value	
[F1]	VK_F5	74	-
[F2]	VK_F6	75	-
[F3]	VK_F7	76	-
[F4]	VK_F8	77	-
[▲]	VK_F9	78	-
[▼]	VK_F11	7A	-
[◀]	VK_F10	79	-
[▶]	VK_F12	7B	-
[9]	-	DB	5B ([])
[8]	-	DE	27 (')
[7]	-	DC	5C (¥)
[6]	-	C0	60 (`)
[5]	-	BD	2D (-)
[4]	-	BF	2F (/)
[3]	-	BB	3D (=)
[2]	-	BA	3B (;)
[1]	VK_SPACE	20	20(SPACE)
[0]	-	DD	5D ())
[.]	-	BC	2C (,)
[SP]	VK_SPACE	20	20(SPACE)
[ESC]	VK_ESCAPE	1B	1B
[SF]	VK_CAPITAL	14	-
[FN]	VK_FUNC	D2	-
[AL]	VK_ALP	D0	-
[BS]	VK_BACK	08	08
[ENT]	VK_RETURN	D0	-
[SCAN]	VK_SCAN	D1	-
[TAB]	VK_SEND	D3	-
[M1]	VK_M1(*1)	C1(*1)	-(*1)
[M2]	VK_M2(*1)	C2(*1)	-(*1)
[M3]	VK_M3(*1)	C3(*1)	-(*1)

(*1) Virtual key codes and character codes will differ based on the key settings.
For details, refer to section "6.4 Magic Key Control"

A.1.3. Character Codes in Alphabet Entry Mode

In the alphabetic entry mode, the 0 to 9 and period (.) keys are used to enter alphabets. The table below lists the relationship between keys to be pressed, the number of depressions, and character codes.

Depre- sion Key	1st	2nd	3rd	4 th	5 th	6 th	7 th	8th	9th
[0]	' '	'/'	' ' (blank)	(*1)					
[1]	'.'	'*'	(*1)						
[2]	'A'	'B'	'C'	'a'	'b'	'c'	(*1)		
[3]	'D'	'E'	'F'	'd'	'e'	'f'	(*1)		
[4]	'G'	'H'	'I'	'g'	'h'	'i'	(*1)		
[5]	'J'	'K'	'L'	'j'	'k'	'l'	(*1)		
[6]	'M'	'N'	'O'	'm'	'n'	'o'	(*1)		
[7]	'P'	'Q'	'R'	'S'	'p'	'q'	'r'	's'	(*1)
[8]	'T'	'U'	'V'	't'	'u'	'v'	(*1)		
[9]	'W'	'X'	'Y'	'Z'	'w'	'x'	'y'	'z'	(*1)
[.]	'.'	'%'	'\$'	(*1)					

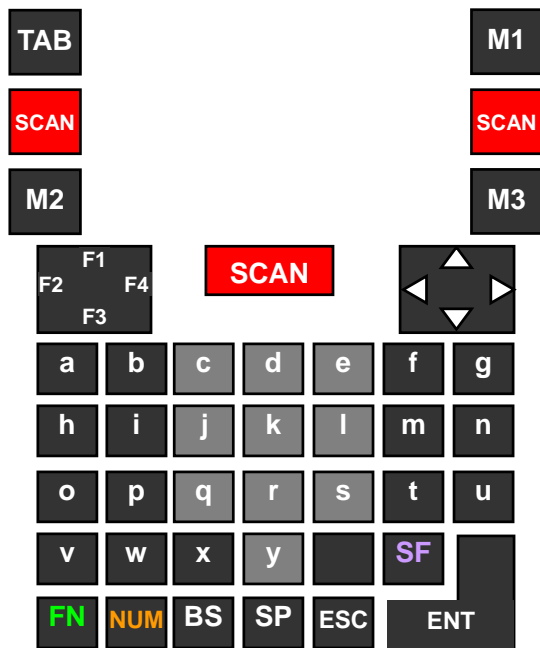
(*1): Returns to the 1st letter.

Character code and virtual key code are notified at establishing character code.

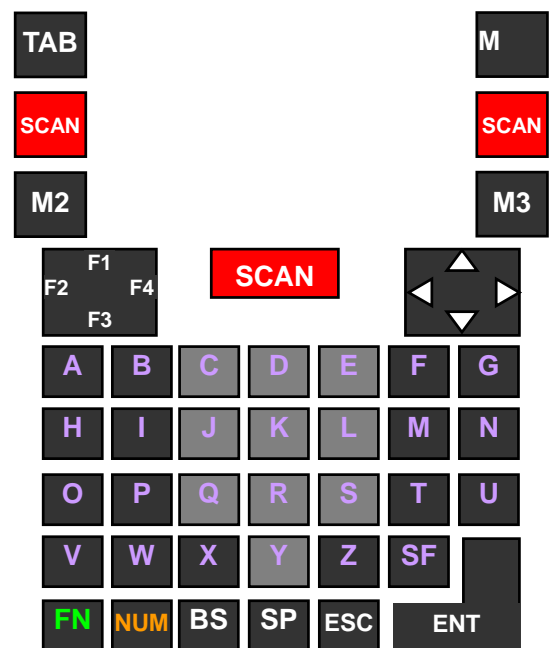
A.2. 42-key pad

A.2.1. Keyboard Arrangement

(1) Alphabet entry mode

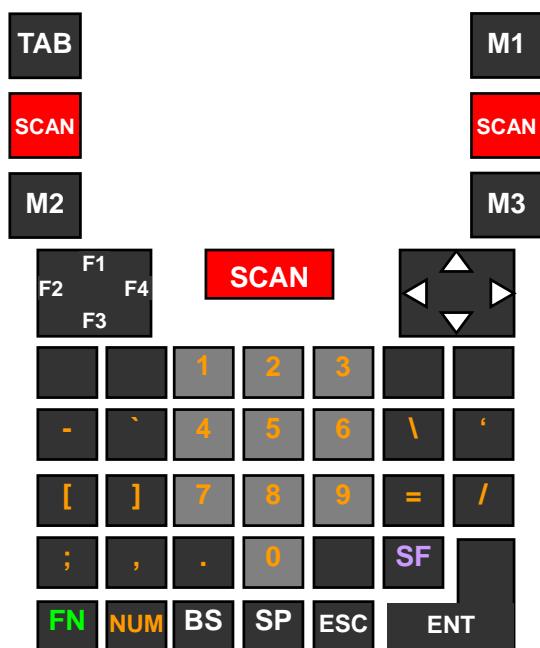


Normal status

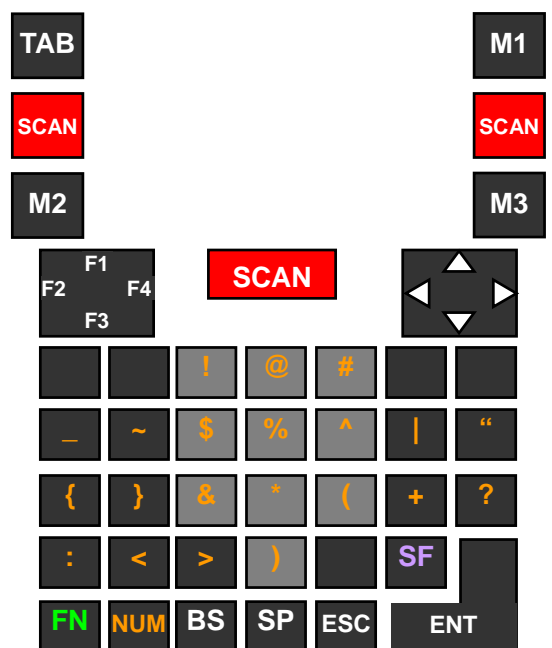


Status with [SF] pressed

(2) Numeric entry mode

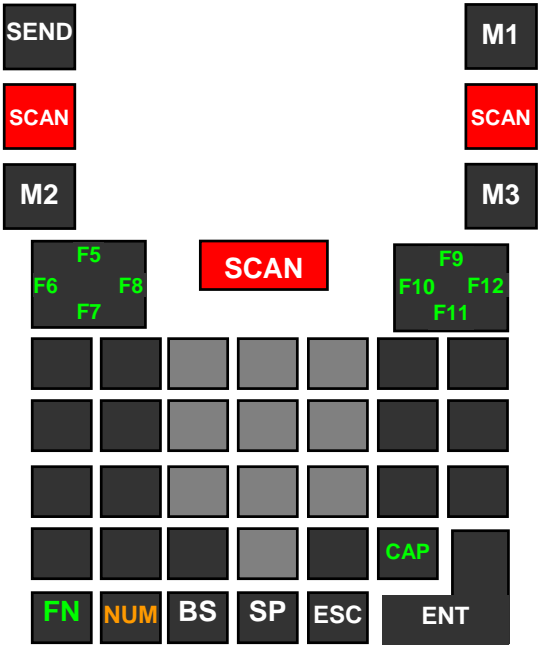


Normal status



Status with [SF] pressed

(3) Function mode



A.2.2. Virtual Key Codes and Character Codes

Alphabet entry mode

Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
	Constant	Value		Constant	Value	
[F1]	VK_F1	70	-	←	←	←
[F2]	VK_F2	71	-	←	←	←
[F3]	VK_F3	72	-	←	←	←
[F4]	VK_F4	73	-	←	←	←
[▲]	VK_UP	26	-	←	←	←
[▼]	VK_DOWN	28	-	←	←	←
[◀]	VK_LEFT	25	-	←	←	←
[▶]	VK_RIGHT	27	-	←	←	←
[A]	VK_A	41	41(A)	←	←	A
[B]	VK_B	42	42(B)	←	←	B
[C]	VK_C	43	43(C)	←	←	C
[D]	VK_D	44	44(D)	←	←	D
[E]	VK_E	45	45(E)	←	←	E
[F]	VK_F	46	46(F)	←	←	F
[G]	VK_G	47	47(G)	←	←	G
[H]	VK_H	48	48(H)	←	←	H
[I]	VK_I	49	49(I)	←	←	I
[J]	VK_J	4A	4A(J)	←	←	J
[K]	VK_K	4B	4B(K)	←	←	K
[L]	VK_L	4C	4C(L)	←	←	L
[M]	VK_M	4D	4D(M)	←	←	M
[N]	VK_N	4E	4E(N)	←	←	N
[O]	VK_O	4F	4F(O)	←	←	O
[P]	VK_P	50	50(P)	←	←	P
[Q]	VK_Q	51	51(Q)	←	←	Q
[R]	VK_R	52	52(R)	←	←	R
[S]	VK_S	53	53(S)	←	←	S
[T]	VK_T	54	54(T)	←	←	T
[U]	VK_U	55	55(U)	←	←	U
[V]	VK_V	56	56(V)	←	←	V
[W]	VK_W	57	57(W)	←	←	W
[X]	VK_X	58	58(X)	←	←	X
[Y]	VK_Y	59	59(Y)	←	←	Y
[Z]	VK_Z	5A	5A(Z)	←	←	Z
[SF]	VK_SHIFT	10	-	←	←	←
[FN]	VK_FUNC	D2	-	←	←	←
[NUM]	VK_NUM	D4	-	←	←	←
[BS]	VK_BACK	08	08	←	←	←
[SP]	VK_SPACE	20	20	←	←	←
[ESC]	VK_ESCAPE	1B	1B	←	←	←
[ENTER]	VK_RETURN	0D	0D	←	←	←
[TAB]	VK_TAB	09	09	←	←	←
[SCAN]	VK_SCAN	D1	-	←	←	←
[M1]	VK_M1	C1(*1)	-(*1)	←	←	←
[M2]	VK_M2	C2(*1)	-(*1)	←	←	←
[M3]	VK_M3	C3(*1)	-(*1)	←	←	←

(*1) Virtual key codes and character codes will differ based on the key settings.

For details, refer to section “6.4 Magic Key Control”

Numeric entry mode

Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
[F1]	VK_F1	70	-	←	←	←
[F2]	VK_F2	71	-	←	←	←
[F3]	VK_F3	72	-	←	←	←
[F4]	VK_F4	73	-	←	←	←
[▲]	VK_UP	26	-	←	←	←
[▼]	VK_DOWN	28	-	←	←	←
[◀]	VK_LEFT	25	-	←	←	←
[▶]	VK_RIGHT	27	-	←	←	←
[A]	-	-	-	←	←	←
[B]	-	-	-	←	←	←
[C]	VK_1	31	1	←	←	21(!)
[D]	VK_2	32	2	←	←	40(@)
[E]	VK_3	33	3	←	←	23(#)
[F]	-	-	-	←	←	←
[G]	-	-	-	←	←	←
[H]	VK_HYPHEN	BD	-	←	←	5F(_)
[I]	VK_BACKQUOTE	C0	`	←	←	7E(~)
[J]	VK_4	34	4	←	←	24(\$)
[K]	VK_5	35	5	←	←	25(%)
[L]	VK_6	36	6	←	←	5E(^)
[M]	VK_BACKSLASH	DC	¥	←	←	7C()
[N]	VK_APOSTROPHE	DE	'	←	←	22(")
[O]	VK_LBRACKET	DB	[←	←	7B({)
[P]	VK_RBRACKET	DD]	←	←	7D(})
[Q]	VK_7	37	7	←	←	26(&)
[R]	VK_8	38	8	←	←	2A(*)
[S]	VK_9	39	9	←	←	28(()
[T]	VK_EQUAL	BB	=	←	←	2B(+)
[U]	VK_SLASH	BF	/	←	←	3F(?)
[V]	VK_SEMICOLON	BA	;	←	←	3A(:)
[W]	VK_COMMA	BC	,	←	←	3C(<)
[X]	VK_PERIOD	BE	.	←	←	3E(>)
[Y]	VK_0	30	0	←	←	29())
[Z]	-	-	-	←	←	←
[SF]	VK_SHIFT	10	-	←	←	←
[FN]	VK_FUNC	D2	-	←	←	←
[NUM]	VK_NUM	D4	-	←	←	←
[BS]	VK_BACK	08	08	←	←	←
[SP]	VK_SPACE	20	20	←	←	←
[ESC]	VK_ESCAPE	1B	1B	←	←	←
[ENTER]	VK_RETURN	0d	0d	←	←	←
[TAB]	VK_TAB	09	09	←	←	←
[SCAN]	VK_SCAN	D1	-	←	←	←
[M1]	VK_M1(*1)	C1(*1)	-(*)	←	←	←
[M2]	VK_M2(*1)	C2(*1)	-(*)	←	←	←
[M3]	VK_M3(*1)	C3(*1)	-(*)	←	←	←

(*1) Virtual key codes and character codes will differ based on the key settings.

For details, refer to section "6.4 Magic Key Control"

Function mode

Key	Normal status		
	Virtual key code		Character code
[F1]	VK_F5	74	-
[F2]	VK_F6	75	-
[F3]	VK_F7	76	-
[F4]	VK_F8	77	-
[▲]	VK_F9	26	-
[▼]	VK_F10	28	-
[◀]	VK_F11	25	-
[▶]	VK_F12	27	-
[A]	-	-	-
[B]	-	-	-
[C]	-	-	-
[D]	-	-	-
[E]	-	-	-
[F]	-	-	-
[G]	-	-	-
[H]	-	-	-
[I]	-	-	-
[J]	-	-	-
[K]	-	-	-
[L]	-	-	-
[M]	-	-	-
[N]	-	-	-
[O]	-	-	-
[P]	-	-	-
[Q]	-	-	-
[R]	-	-	-
[S]	-	-	-
[T]	-	-	-
[U]	-	-	-
[V]	-	-	-
[W]	-	-	-
[X]	-	-	-
[Y]	-	-	-
[Z]	-	-	-
[SF]	VK_CAPITAL	14	-
[FN]	VK_FUNC	D2	-
[NUM]	VK_NUM	D4	-
[BS]	VK_BACK	08	08
[SP]	VK_SPACE	20	20
[ESC]	VK_ESCAPE	1B	1B
[ENTER]	VK_RETURN	0d	0d
[TAB]	VK_TAB	09	09
[SCAN]	VK_SCAN	D1	-
[M1]	VK_M1(*1)	C1(*1)	-(*)
[M2]	VK_M2(*1)	C2(*1)	-(*)
[M3]	VK_M3(*1)	C3(*1)	-(*)

(*1) Virtual key codes and character codes will differ based on the key settings.
For details, refer to section “6.4 Magic Key Control”

Appendix B. Differences between Older Unit

The following table lists differences between the BHT-700 and the BHT-400.

Type	Item	BHT-700	BHT-400
System information	ROM capacity	128MB	64MB
Screen display	Rotation function	Possible	Not possible
Backlight	Key backlight	Equipped	Not equipped
	Key backlight illumination trigger	Key press Touch panel tap	No key backlight
	Backlight control function assigned to [Fx](x:1-4) key or [SF] + [Fx] keys.	Not possible	Possible
	Backlight control function assigned to [SCAN] key or [SF] + [SCAN] keys.	Not possible	Possible
Sound	Audio output	Earphone jack, receiver	None
	Audio input	Microphone	None
	Laser key click sound	No receiver key	Click sound ON/OFF possible.
	Half-press key click sound	No half-press key	Volume change possible.
	MessageBox, MessageBeep, PlaySound voice output destination	Earphone jack Receiver	Beeper
	VoIP	Supported	Not supported
Keyboard	No. of keys	27-key, 42-key	30-key, 50-key
	Magic keys	M1, M2, M3	M1, M2, M3, M4, M5
	Laser key	Not supported	Magic key assignment possible.
	Functions assignable to [SCAN] key	Trigger (fixed)	Same as magic keys
	Default alphabet case	Lower case	Upper case Default : Switching possible with CAPS mode.
	[F5] to [F12] independent entry	[F1] to [F4] key or arrow key press in Function mode	[F1] to [F4] key or arrow key press in Function mode by full function assignment
	Terminal service support	Standard U.S. version keypad	BHT original keypad Switching possible with keyboard emulation function.
	Handle attachment	Not supported	Support possible
Icons	Numeric entry icon	Display/hidden switching possible.	Not supported
	SIP icon	BHT original	Windows CE standard + BHT original Default: Windows CE standard only
Power management	Auto power OFF when wireless communication open	Changed with BHT_SetSysSettingDW (BHT_PM_SUSPEND_RF,...).	Changed with BHT_SetSysSettingDW (BHT_PM_SUSPEND_SLOT0,...)
LED	Red LED	Independent charge LED and indicator LED	OR connection for charge LED and indicator LED

Type	Item	BHT-700	BHT-400
Data communication	Default baud rate	115200 bps	9600 bps
	RS-232C	Tx, Rx	Tx, Rx, RTS, CTS
	I/F usable with Active Sync	USB, IrDA	USB, IrDA, RS-232C
	Automatic connection	Placement in cradle connected to computer by USB (Default : Allow)	Connection of cable connecting BHT with computer (Default : Prohibit)
	USB-LAN communication	Supported	Not supported
Wireless	Wireless system	Selection from 802.11a, 802.11b, 802.11b/g (Default: 802.11b)	802.11b/g
	WPA2	Supported	Not supported
Barcode reading (1D)	RSS	Reading possible	Reading possible (Not possible in models for the domestic Japanese market.)
	Marker mode	Not equipped with marker	Selection from Normal, Ahead or No illumination
Code reading (2D)	CODE-93	Reading possible	Reading not possible (BHT-202Q)
	Illumination mode	Selection from Auto, Always ON, or Always OFF	Not supported
OS update	Procedure	Calling the BHT_SysModify function after saving the OS file to a suitable location	Calling the BHT_ShutdownSystem function and ensuring an area to store the OS file, and then calling the BHT_SysModify function after saving the OS file to the secured area.
	Update file back-up location	Arbitrary	Sysmodify folder, CF slot
Equipped device	BHT_GetDeviceInfo	Barcode, COM, ActiveSync compatible device acquisition	Not supported
Touch panel	Attachment status acquisition	Not possible	Possible
ScannerXXXQ.ocx	ReadType property default value	Q:E, A, I:4-99, L:1-99, M:1-99, N:3-99, K:1-99, Y, X, Z, R, V	Q:E, A, I:4-99, M:1-99, N:3-99, K:1-99, Y, X, Z, R, V (BHT-202Q)
FileTransfeXXX.ocx	Default baud rate	115200 bps	9600 bps

BHT-700-CE API Reference Manual

First Edition, October 2007
DENSO WAVE INCORPORATED

The purpose of this manual is to provide accurate information in the development of application programs for the BHT-700. Please feel free to send your comments regarding any errors or omissions you may have found, or any suggestions you may have for generally improving the manual.

In no event will DENSO WAVE be liable for any direct or indirect damages resulting from the application of the information in this manual.